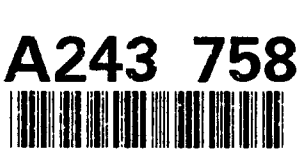


SIMULATION OF FINITE-PRECISION EFFECTS IN DIGITAL FILTERS

THESIS

Perry L. Choate
Captain, USAF

AFIT/GE/ENG/91-D-11

AFIT/GE/ENG/91-D-11

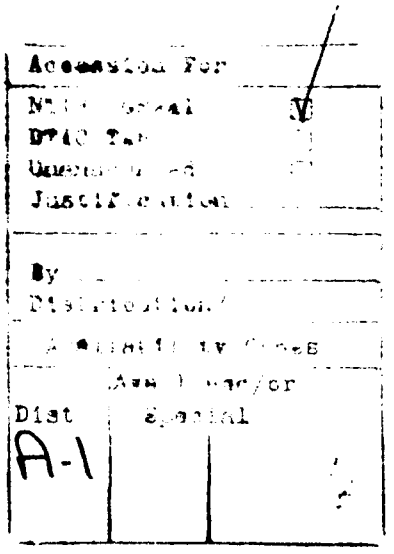
SIMULATION OF FINITE-PRECISION EFFECTS IN DIGITAL FILTERS

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Perry L. Choate, B.S.E.E.
Captain, USAF

December 12, 1991



Approved for public release; distribution unlimited

Acknowledgments

My sincere gratitude and appreciation goes to those who took the extra time to teach me while I was involved in the exciting world of attaining a master's degree at AFIT. The tools I used allowed me to stay productive. The computer work stations helped immensely in the task of creating a software tool efficiently. I was able to create and subdue 2500 lines of code into a presentable framework with great dispatch. Furthermore, since I appeared on the disk drive 'hog' list, special treatment was given to me by the system administrator by moving me to my own private area with lots of extra room.

I appreciate Martin Desimio, my thesis advisor, who provided to me his continued support throughout the thesis process. His time was quite valuable since he himself was attending classes and finishing up his Ph.D. His remarks like, "Work, work, work." and "Work harder." unceasingly kept me from getting too serious and pulling those all nighters. Martin Desimio was always willing to spend extra time during the development process to help form the product. I owe him a debt for learning so much from him personally.

I was able to create a secondary study track in discrete signals. Capt Rob Williams came back to AFIT from FTD to teach me Adaptive Filters. Even he spent some Saturdays with me so that I would develop a feel for the material.

When I arrived at AFIT in the spring of 1990, my family (my lovely wife Susan, and three boys then with ages one, four, and seven) were ready to support me during long days, and evenings studying. Then Dave was born during the middle of AFIT on 20Jan91. The amount of disorganization invented daily for Susan to wrest was a task for two able bodied adults. The thanks will always be in my heart for her hard work with the home and the children. I love you Susan and all four boys.

Perry L. Choate

Table of Contents

Page

Acknowledgments ii

Table of Contents iii

List of Figures vii

List of Tables xiii

Abstract xiv

I. **Introduction** 1-1

1.1 Background 1-1

1.2 Problem Statement 1-2

1.3 Computer Simulation of Digital Filters 1-2

1.4 Thesis Objectives 1-3

1.5 Assumptions 1-3

1.6 Scope 1-3

1.7 Methodology 1-4

1.8 Benefits of the Research 1-5

1.9 Equipment 1-5

1.10 Thesis Organization 1-6

II. **Background** 2-1

2.1 Noise Sources 2-1

2.2 Discrete-Time Filter Design 2-1

2.2.1 Difference Equations. 2-2

Page

2.2.2 The z-Transform. 2-2

2.2.3 Quantization Errors for Digital Filters. 2-4

2.3 Roundoff Noise in Digital Filters 2-10

2.3.1 Modeling Roundoff Error In Cascaded Filter Sections. . 2-10

2.4 Summary 2-12

III. **Theory of Finite-Precision Effects** 3-1

3.1 Overview 3-1

3.2 Realization of Difference Equations 3-1

3.2.1 Filter Forms 3-1

3.2.2 Arranging the Difference Equation. 3-2

3.2.3 Direct Form I. 3-4

3.2.4 Direct Form II. 3-4

3.2.5 Cascading Second Order Direct Form II Sections. . . . 3-9

3.3 Finite-Precision Considerations for the Digital Filter 3-9

3.3.1 Finite-Precision Representation. 3-9

3.3.2 Floating-Point Notation Used by the Simulator Tool. . 3-11

3.3.3 Fixed-Point Notation Used By The Digital Filter Simulator Tool. 3-11

3.4 Quantization Error 3-12

3.4.1 Pole-Zero Locations 3-13

3.4.2 Quantization of Singularities. 3-15

3.4.3 Sensitivity of Pole/Zero Locations. 3-16

3.5 Linearized Models For Roundoff Errors 3-18

3.5.1 Signal-to-Quantization Noise 3-19

3.5.2 Modeling Of Roundoff Noise. 3-20

3.5.3 Roundoff Power Computation. 3-24

Page

IV. **Digital Filter Analysis Tool** 4-1

4.1 Overview of the Digital Filter Analysis Tool 4-1

4.2 The Main Computation Section for Computations 4-1

4.2.1 Calculation of Unquantized Magnitude and Phase . . . 4-2

4.2.2 Calculation of Quantized Magnitude and Phase 4-2

4.3 Subroutines Called by the Main Option Menu and Main Computation Section 4-3

4.3.1 Function Quant; Performs the Quantization and Roundoff Computations 4-3

4.3.2 Roundoff Power Computations 4-4

4.3.3 Subroutine ChangeCascade Order; Allows Manipulation of the Cascade Sections for the Numerator and the Denominator. 4-8

4.3.4 Summary 4-11

V. **Experimental Results From The Digital Filter Simulator Tool** . 5-1

5.1 Overview of Chapter 5-1

5.2 Comparison Of Roundoff Power Computations. 5-1

5.3 FIR Example With Linear Phase Showing Results with Coefficients Quantized. 5-3

5.4 FIR Example Implemented in Direct and Cascade Form 5-15

5.5 IIR Bandpass Example Implemented in Direct Form to Show Robustness in IIR Filter 5-25

5.6 Cascade Bandpass Filter, Eighth-Order, Four Sections 5-33

5.7 Cascade Filter, Twelfth-Order 5-40

5.8 Summary 5-55

VI. Conclusions and Recommendations For Further Study 6-1

6.1 Conclusions 6-1

Page

6.1.1 Coefficient Representation 6-1

6.1.2 Analysis of Results 6-1

6.2 Recommendations For Further Study 6-2

Appendix A. Program Listing for Digital Software Analyzer Tool A-1

Appendix B. User's Manual To Use The Digital Software Analyzer Tool . . . B-1

B.1 The Input Needed And The Output Generated B-1

B.2 How The Digital Filter Analysis Tool is Organized B-2

B.2.1 The Controlling Section B-2

B.2.2 Direct Structure Required to Build Coefficient Files . B-3

B.2.3 Cascade Structure Required to Build Coefficient Files B-5

B.3 Controlling Section, Main Menu Options To Run The Digital Analyzer B-7

Appendix C. Additional Subroutines for the Digital Filter Analysis Software Tool C-1

C.1 Inputs to the Digital Filter Analysis Software Tool C-1

C.1.1 Subroutines for Structual Evaluation C-6

C.1.2 Sign Convention for Specification of Coefficients. C-9

Vita VITA-1

Bibliography BIB-1

List of Figures

Figure Page

2.1. Direct Form 1 IIR digital filter. 2-5

2.2. Second order sections cascaded in Direct Form II. 2-6

2.3. Linear noise modeling for direct form I showing the injection of noise sources after each multiplication. 2-7

2.4. Linear noise modeling for direct form I showing superposition of noise sources into one noise source injected into the system. 2-8

2.5. Model to calculate the output of a transfer function to include the effects due to coefficient quantization shown as $\Delta H(z)$. 2-10

3.1. Example block diagram for a second-order difference equation. 3-3

3.2. Block diagram for the general order difference equation in direct form I. . 3-5

3.3. Block diagram for the general order difference equation in direct form II. . 3-7

3.4. Reducing the delay elements to a minimum for the canonic form and the second-order canonic form. 3-8

3.5. Fourth-order filter and an eighth-order filter realized with cascaded second-order sections. 3-10

3.6. Allocation of bits in a fixed-point two's complement representation (16-bit word shown). 3-12

3.7. Number quantization effects with 4-bits, Two's complement rounding. . . 3-14

3.8. Direct form implementation for a complex-conjugate pole pair. 3-16

3.9. Possible locations for singularities of a second-order FIR direct form implementation. 3-17

3.10. Linear noise modeling for direct form I showing the injection of noise sources after each multiplication. 3-21

3.11. Linear noise modeling for direct form I showing the summing of noise sources into one noise source. 3-22

Figure Page

4.1. Cascade structure with Direct Form II realization for the second-order section. 4-6

4.2. Recursive Mean Estimator Implementation 4-7

4.3. Infinite Impulse Response structure, second-order feedback and second order feedfoward sections. 4-8

4.4. Pole-zero plot for a fourth-order filter showing pairing of most reasonable pairs. 4-9

5.1. Unquantized Magnitude Plot, Full Single Precision, representation of coefficients in Table 5.3. 5-5

5.2. Quantized Magnitude Plot, 16-Bits Precision, representation of coefficients in Table 5.3. 5-5

5.3. Quantized Magnitude Plot, 14-Bits Precision, representation of coefficients in Table 5.3. 5-6

5.4. Quantized Magnitude Plot, 13-Bits Precision, representation of coefficients in Table 5.3. 5-6

5.5. Quantized Magnitude Plot, 12-Bits Precision, representation of coefficients in Table 5.3. 5-7

5.6. Quantized Magnitude Plot, 10-Bits Precision, representation of coefficients in Table 5.3. 5-7

5.7. Quantized Magnitude Plot, 8-Bits Precision, representation of coefficients in Table 5.3. 5-8

5.8. Quantized Magnitude Plot, 6-Bits Precision, representation of coefficients in Table 5.3. 5-8

5.9. Linear error (difference from the single precision version), 16-Bits precision, representation of coefficients in Table 5.3. 5-9

5.10. Linear error (difference from the design criteria) 16-Bits precision, pass band and stop band shown, representation of coefficients in Table 5.3. 5-9

5.11. Linear error (difference from the single precision version), 12-Bits precision, representation of coefficients in Table 5.3. 5-10

Figure Page

5.12. Linear error (difference from the design criteria), 12-Bits accuracy, pass band and stop band shown,representation of coefficients in Table 5.3. . . . 5-10

5.13. Linear error (difference from the single precision version), 8-Bits accuracy, representation of coefficients in Table 5.3. 5-11

5.14. Linear error (difference from the design criteria), 8-Bits accuracy, representation of coefficients in Table 5.3. 5-11

5.15. Phase Response 16-Bits accuracy, representation of coefficients in Table 5.3. 5-12

5.16. Phase Response, 14-Bits precision, representation of coefficients in Table 5.3. 5-12

5.17. Phase Response, 12-Bits precision, representation of coefficients in Table 5.3. 5-13

5.18. Phase Response, 10-Bits precision, representation of coefficients in Table 5.3. 5-13

5.19. Phase Response, 8-Bits precision, representation of coefficients in Table 5.3. 5-14

5.20. Single precision Magnitude Plot, conversion from cascade structure to direct structure, representation of coefficients in Table 5.7. 5-19

5.21. Quantized Magnitude Plot, direct structure 16-Bits precision, representation of coefficients in Table 5.5. 5-20

5.22. Quantized Magnitude Plot, direct structure 8-Bits precision, representation of coefficients in Table 5.5. 5-20

5.23. Quantized Magnitude Plot, cascade structure 16-Bits precision, representation of coefficients in Table 5.6. 5-21

5.24. Quantized Magnitude Plot, cascade structure 8-Bits precision, representation of coefficients in Table 5.6. 5-21

5.25. Magnitude Plot, conversion from cascade structure to direct structure using 16-bits precision, representation of coefficients in Table 5.7. 5-22

5.26. Linear Magnitude Error with direct structure, 16-Bits precision, representation of coefficients in Table 5.5. 5-22

5.27. Linear Magnitude Error with direct structure, 8-Bits precision, representation of coefficients in Table 5.5. 5-23

5.28. Linear Magnitude Error with cascade structure, 16-Bits precision, representation of coefficients in Table 5.6. 5-23

Figure Page

5.29. Linear Magnitude Error with cascade structure 8-Bits precision, representation of coefficients in Table 5.6. 5-24

5.30. Linear Magnitude Error Plot, conversion process from cascade to direct, 16-bits precision, representation of coefficients in Table 5.7. 5-24

5.31. 24-bit Magnitude Plot, representation of coefficients in Table 5.8. 5-26

5.32. Quantized Magnitude Plot, 16-Bits precision, representation of coefficients in Table 5.8. 5-26

5.33. Quantized Magnitude Plot, 12-Bits resolution, representation of coefficients in Table 5.8. 5-27

5.34. Quantized Magnitude Plot, 8-Bits resolution, representation of coefficients in Table 5.8. 5-27

5.35. Quantized Magnitude Plot, 6-Bits resolution, representation of coefficients in Table 5.8. 5-28

5.36. Linear Magnitude Error, 16-Bits resolution, representation of coefficients in Table 5.8. 5-28

5.37. Linear Magnitude Error, 12-Bits resolution, representation of coefficients in Table 5.8. 5-29

5.38. Linear Magnitude Error, 8-Bits resolution, representation of coefficients in Table 5.8. 5-29

5.39. Linear Magnitude Error, 6-Bits resolution, representation of coefficients in Table 5.8. 5-30

5.40. Phase Response, 16-Bits resolution, representation of coefficients in Table 5.8. 5-30

5.41. Phase Response, 12-Bits resolution,, representation of coefficients in Table 5.8. 5-31

5.42. Phase Response, 8-Bits resolution, representation of coefficients in Table 5.8. 5-31

5.43. Phase Response, 6-Bits resolution, representation of coefficients in Table 5.8. 5-32

5.44. 24-bit single precision Magnitude Plot, cascade structure, representation of coefficients in Table 5.9. 5-35

5.45. Quantized Magnitude Plot, 16-Bits precision, cascade structure, representation of coefficients in Table 5.9. 5-36

Figure Page

5.46. Quantized Magnitude Plot, 12-Bits precision, cascade structure, representation of coefficients in Table 5.9. 5-37

5.47. Quantized Magnitude Plot, 8-Bits precision, cascade structure, representation of coefficients in Table 5.9. 5-37

5.48. Linear Magnitude Error with 16-Bits for Coefficients, cascade structure, representation of coefficients in Table 5.9. 5-38

5.49. Linear Magnitude Error with 12-Bits for Coefficients, cascade structure, representation of coefficients in Table 5.9. 5-38

5.50. Linear Magnitude Error with 8-Bits for Coefficients, Cascade structure, representation of coefficients in Table 5.9. 5-39

5.51. 24-bit Magnitude Plot, cascade structure representing coefficients in Table 5.11, column two. 5-45

5.52. 24-bit Magnitude Plot, cascade structure representing coefficients in Table 5.11, column two. 5-46

5.53. Magnitude Plot, 16-Bits normalized cascade representing coefficients in Table 5.10 5-46

5.54. Magnitude Plot, 16-Bits normalized cascade representing coefficients in Table 5.10. 5-47

5.55. Magnitude Plot, 8-Bits normalized cascade representing coefficients in Table 5.11. 5-47

5.56. Magnitude Plot, 8-Bits normalized cascade representing coefficients in Table 5.11. 5-48

5.57. Phase Response 24-bit Coefficients, cascade structure representing coefficients in Table 5.11. 5-48

5.58. Phase Response 24-bit Coefficients, cascade structure representing coefficients in Table 5.11. 5-49

5.59. Phase Response with 16-Bits for Coefficients, cascade structure representing coefficients in Table 5.10. 5-49

5.60. Phase Response with 16-Bits for Coefficients, cascade structure representing coefficients in Table 5.10. 5-50

Figure Page

5.61. Phase Response with 8-Bits for Coefficients, cascade structure representing coefficients in Table 5.11. 5-50

5.62. Phase Response with 8-Bits for Coefficients, cascade structure representing coefficients in Table 5.11. 5-51

5.63. Magnitude Plot, 8-bits to represent coefficients in Table 5.12, normalized conversion from the cascade stru' ure, direct form I. 5-52

5.64. Magnitude Plot, 16-bits to represent coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I. 5-53

5.65. Magnitude Plot, 24-bit Single Precision coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I. 5-53

5.66. Magnitude Plot, 48-bit Double Precision fo. coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I. 5-54

B.1. Main Menu Options Screen for the Digital Filter Simulator B-8

C.1. Two types of structures analyzed by the digital software analyzer tool. . . C-4

List of Tables

Table Page

3.1. Representation of numbers with 2-bits. 3-12

3.2. Representation of numbers with 3-bits. 3-13

3.3. Roundoff power measurements for different wordlengths computed by theoretical means. 3-26

5.1. Coefficients for the Cascade section used for roundoff power calculation comparisons. 5-2

5.2. Comparison of roundoff power measurements for different wordlengths computed by theoretical means and by the software simulator. 5-2

5.3. Unquantized and quantized coefficients for an FIR filter. 5-3

5.4. Coefficients to Begin the Investigation of an FIR Filter. 5-15

5.5. Coefficients in Non-normalized and Normalized States for the FIR Filter In Direct Form I. 5-16

5.6. Coefficients for the cascade second-order sections before and after normalization. 5-17

5.7. Coefficients in direct form before and after the conversion process for an FIR Filter. 5-18

5.8. Coefficients for IIR Bandpass Example. 5-25

5.9. Unquantized (single precision) and quantized coefficients for an IIR Cascade filter. 5-34

5.10. Unquantized and quantized coefficients for an IIR Cascade filter. 5-41

5.11. Coefficients for the Cascade structure and the results of the normalization. 5-43

5.12. Results of the Conversion process from Non-Normalized coefficients and Normalized coefficients; the resulting direct structure's coefficients are then normalized. 5-44

C.1. Forms of the difference equation in time, z-transform, and frequency. . . . C-7

Abstract

This thesis develops and demonstrates software that models finite-precision effects in digital filters. Coefficient quantization and internally generated roundoff noise effects are simulated by rounding coefficients and multiplier output values to (B+1) bits. Outputs include magnitude and phase responses that may be plotted to show results of coefficient quantization. The output noise power due to roundoff is calculated.

Three types of digital filter structures are examined. The first type is direct form I implementation for FIR and IIR structures, second is direct form II, and third is cascading the canonic direct form II second-order sections for higher order filter designs. Re-arrangement of the second-order sections can be done to examine the change in the roundoff power generated. One other feature converts the cascade canonic form into a direct form for analysis. The software is menu driven with help screens written in FORTRAN77. Options include managing output files and tailoring the simulation results. Examples are given to show that the simulation provides accurate results. Theoretical results are developed and compare favorably to the simulation output for verification.

SIMULATION OF FINITE-PRECISION EFFECTS IN DIGITAL FILTERS

I. Introduction

Research in the area of digital filtering systems increased dramatically in the early 70's with the availability of the microprocessor. Previous work done in the area of digital filters peaked during this time when microprocessors were in their infancy. At this time, the theory to model internal noise sources formed [12, 7, 4]. However, techniques to model digitally generated noise sources takes on new flavors. Many new formats now exist for the representation of numbers. Some formats are unique to the manufacturer and other formats are defined by IEEE. In addition, the increased width of the data bus path provides a mechanism to attenuate much of the noise problems associated with coefficient quantization and roundoff noise generation in digital filters from addition and multiplication.

1.1 Background

In 1965 the Cooley-Tukey algorithm established a new mechanism to perform Fourier analysis in a very fast routine [1:297-301]. This algorithm provided to the signal processing engineer an efficient mechanism to compute the frequency content of a signal. The bi-linear transform also made significant contributions to the signal processing engineer during this time period. The bi-linear transform provided a mechanism to derive the digital equivalent of the analog filters [9:221-236]. These powerful tools boosted the importance of digital signal processing.

This thesis effort continues research in the area of finite-precision effects. Performance criteria drives how accurate the filter needs to be, but digital filter implementation has some difficulties. Two basic problems exist in the realization of digital filters. Coefficient quantization effects and roundoff noise sources result from implementing digital filters. Linear time-invariant discrete-time systems are most common to implement the filter operation, so this research will focus on the realization of these types of digital filters. The two structures used are the direct form and the cascade form.

Performance criteria drives how accurate the filter needs to be. Digital filters are designed with constants like π with theoretically infinite accuracy. The coefficients or weights used in the digital filter are also designed as irrational numbers. The weights are used as multipliers within the filter structure. Computers can only provide a finite number of bits to represent digital data. Thus, truncating of the weights in the designed filter coefficients results in a process called coefficient quantization. Since designers are interested in implementations requiring the most simplistic design in software and hardware, the effects of coefficient quantization must be studied.

Results from the multiplication and addition used within the filter structure are often numbers requiring more digits of accuracy than the multiplicants used for the multiplication process. When a B-bit length number is multiplied by another B-bit length number, the result will require a 2B-bit length number to hold the result. However, the result is truncated back into a B-bit data value. This process of truncating the results of multiplication is called roundoff noise error.

Another problem results from the form that the digital filter takes. Different structures of the same filter will result in different performance results. Often digital filters are designed by lumping together second-order sections. This process is called second-order cascade design when cascade structures are used. The second-order cascade sections are generally more robust than the direct form. The degradation in performance due to structural differences is due to both coefficient quantization and roundoff noise.

1.2 Problem Statement

Implementations of digital filters suffer from the effects of coefficient quantization and roundoff noise. This research seeks to build a software tool to observe the effects of the unwanted characteristics of coefficient quantization and roundoff noise power of digital filters.

1.3 Computer Simulation of Digital Filters

To simulate a digital filter on a computer is a natural process. The digital filter itself is a discrete system stepping through iterations. Each iteration provides one output from the computer model. The simulation is stopped after each iteration for additional processing of statistical data and then the computer model is started again. The computer simulation allows the research to progress very quickly since each digital filter can be simulated without actually building the digital signal processing hardware.

The results of the digital filter implemented by the simulation are compiled in files for display. The magnitude plot of the frequency range can be compared to the design performance specifications. After the comparison, the designer is able to make conclusions about the realized filter's performance and if the application will need further modification.

1.4 Thesis Objectives

The objective of this thesis effort is to design and realize a general purpose computer program useful for simulating the effects of coefficient quantization and finding noise performance of different filter structures.

The simulation package to model the effects of internally generated noise sources uses existing theory relating to the implementation of digital filters and internally generated noise sources. The simulator will include options to specify the number of bits to represent coefficients. This will allow a designer to include the accuracy of the coefficients.

1.5 Assumptions

Computer simulations are done with single precision numbers using the IEEE single precision format. Care is taken not to exceed the bounds on the magnitudes available to represent both intermediate and final results. The software simulator will catch division by zero conditions. The structured programming language Fortran77 will provide the user interface necessary for all input/output information and file access.

This research assumes a limit on the number of bits available when the filter is realized to 24-bits. The simulator will be able to handle up to 32-bits for implementing the filter using the Sparc station 2 workstations. The magnitude of the results from the math operations performed within the filter are limited to the IEEE 32-bit floating format.

Noise sources are modeled as uncorrelated, independent random processes with uniform probability density functions. The error due to internal rounding is modeled by a uniform probability density function over the range of the smallest quantization step.

1.6 Scope

Limitations apply to this research effort. The software simulator will be written to provide results on two structures, Direct form I and Direct form II. The Direct form II is limited to second-order sections. The filters will be constant coefficient designs.

There are two basic structures of digital filters. Both forms will be able to simulate Finite Impulse Response (FIR) filters and Infinite Impulse (IIR) filters. The FIR filter has a limited response to one impulse as input to the filter. The IIR will continually produce output from just one impulse into the input of the filter.

The software simulator will take coefficients for the desired filter and perform calculations to show the finite-precision effects. Internally generated noise sources from implementing digital filters affect the transfer function of the filter from a theoretical design perspective. The results are to show changes in the transfer function of the filter and perform roundoff power calculations.

1.7 Methodology

The first objective will involve the understanding necessary to describe the coefficient quantization effects on a digital filter. A simulation of the filter will provide the results from the implemented transfer function that describes the filter. The amount of error resulting from the quantization noise will be related to the movement of poles and zeros that describe the transfer function in the z-plane.

The second objective will involve learning and developing theory necessary to model roundoff noise within digital filters. All digital filters will have limited registers to hold the binary representation resulting from addition and multiplication. Each point in the digital filter that contains a multiplier will have a source of roundoff noise. The roundoff noise source will model the appropriate mathematical operation performed at the node. The operation is pre-determined by the design process and the desired type of implementation for the filter.

The third objective will involve modeling the effects of filter structure when second-order sections are cascaded in a digital filter. More often than not, the performance requirements given to the filter will require the use of more than a second-order filter. These higher order filters are built from cascaded second-order sections. The ordering of these second-order sections will affect the overall noise performance of the filter. A haphazard approach ultimately results in less than desired performance realizations. The simulator will need to describe the effects of cascaded second-order sections on the roundoff noise generated. A roundoff power figure in watts will be found for the filter under study. Furthermore, the user will be able to move the order of the cascaded sections to observe the affect on roundoff power.

The fourth objective will entail the design and realization of a general purpose simulator useful for finding the effects of quantizing the coefficients and evaluating the noise

performance of different filter structures. A computer based model will allow a designer to measure the effects on a specified realization. The software model will be a tool, usable by a menu driven option system. The simulator must have the capability to input the filter coefficients. The filter structure is determined by the user. The results will be the response of the filter in magnitude, phase, and linear error from the theoretical magnitude response. The required number of bits can then be found for the representation of filter weights and lengths of internal storage registers in order to maintain specified performance.

1.8 Benefits of the Research

First, this thesis provides a convenient way to show finite-precision effects. Comparable software packages are more general and do not show these finite-precision effects directly.

Second, this research allows the Air Force to have an engineer skilled in the design of digital filters for applications. Demand to implement filters increases in areas of ground support equipment, air platforms, and space platforms.

Third, this thesis allows a blending of design needs between the digital communication area and the Very Large Scale Integration (VLSI) area. Requirements for each of these two design groups don't meet. The VLSI area requires the structure of a filter using the least amount of complexity while the communication area provides theoretical designs without simulating the realization. Results of this work minimize the problems when designs are passed from one group to the other. Furthermore, this work appropriates sure mechanisms to allow each group to express their specific design criteria.

Lastly, this research provides to the designer a concise treatment of the problems found within digital filter realizations. This thesis grants to the designer a mathematical model to avoid very costly mistakes, assuring sound designs and trouble free realizations.

1.9 Equipment

The equipment used for the research is primarily a Sun computer work station that has a compiler for fortran code and a PC/AT. The PC/AT is used to develop and edit text for the thesis. Sun workstations provide the graphic tools for the thesis and the publisher to create this document.

1.10 Thesis Organization

This thesis is organized into six chapters. Chapter I provides an introduction to the thesis. The problem is stated and some background is expressed. Chapter II is the background review. Specific articles in this topic area are reviewed and conclusions given. Chapter III explains each step in the methodology used to build the theory and applicability of methods to solve the problems. The theory of discrete time signals is reviewed as it applies to the thesis. Chapter IV covers the program that performs the analysis on the filter structure. The sections of code are explained so the user can understand the algorithms designed to implement each task. Chapter V reveals the results from the software tool and verifies the software tool's output. Examples are shown to verify different parts of the software. Chapter VI provides conclusions based on the research and addresses areas for further research. Appendices include the code developed, explainations of subroutines, and a users manual.

II. Background

In this chapter, background is given on the effects of finite-precision effects. Quantifying these effects is done by injection of noise sources into the filter and finding the resulting power. Theory needed to perform the calculations include difference equations and the z-transform. Differences in the Direct Form I and Direct Form II are presented. Coefficient quantization is examined in the filter structure as well as roundoff error. These two forms of finite-precision effects are explored by use of theory.

2.1 Noise Sources

With discrete-time signal processing, a few areas need direct attention. Discrete-time signal processing involves the use of discrete-time filters. These discrete filters, also known as digital filters, have some inherent areas of errors. The following errors exist:

- quantization errors occur during the transformation of continuous-amplitude signals to discrete-amplitude signals
- quantization errors occur in the process of representation of the desired weights in the digital filter and
- roundoff errors occur after arithmetic multiplication computations.

Digital filters can be designed without consideration of the quantization and roundoff noise introduced by implementation. Filter designs use a variety of methods to develop and implement the filter in hardware. The results of the design may or may not prove acceptable.

The designer needs to know how to determine the minimum number of bits required for the filter implementation. Trade-offs between the number of bits and design performance can be made to produce economical and practical implementations.

2.2 Discrete-Time Filter Design

In order to analyze the noise introduced within the digital filter, certain tools that provide simulation of finite-precision effects must be used to focus in on the changes in frequency response. In addition, digital filters can be placed into two types: finite impulse

response and infinite impulse response. The research will involve the use of both finite and infinite impulse response filters.

The reader is directed to texts [3, 4, 15] for background with the topics of difference equations and the z-transforms. These tools allow the design engineer to analyze systems during the design process. The analysis can also show where the most critical sections will be during realization. Areas in the realization that will cause the most deviation in the magnitude response can be found.

2.2.1 Difference Equations. Difference equations embody the concepts of discrete values of time where the filter will perform computations resulting in an output value [8:28-36]. Digital filters receive input sequence and produces an output sequence. Consider the difference equation given by the general form as

$$\sum_{k=0}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]. \tag{2.1}$$

The general form can be rearranged as

$$y[n] = \sum_{k=1}^{N} -a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]. \tag{2.2}$$

where the coefficients are normalized by the value of a_0 in Equation 2.2.

The system that is represented by this general form has the input sequence $x[n]$ and the output sequence $y[n]$. Delayed input and output values are represented by the integer value, k.

The difference equations used are causal in that only present and past values can be represented in the system. The coefficients are constant and the equation is linear. Feedback terms present in the difference equation raise stability issues in the design process. In general, the use of feedback terms causes the filter to have an infinite impulse response. When no feedback terms are used, the filter has a finite impulse response [15:206-213].

2.2.2 The z-Transform. The z-transform is related to the difference equation by a substitution process. The z-transform analysis is applied to the discrete-time signals using the counter k for discrete points in time. The z-transform is a complex transform. The

variable z is a complex variable with properties that are useful to transform from continuous-time domain to discrete-time domain signals[3:194-213].

The z-transform is written with the following form

$$X(z) = \sum_{k=-\infty}^{+\infty} x[k]z^{-k}. \tag{2.3}$$

The z-transform of the difference equation shown in Eq. 2.2 can be written as

$$\frac{Y(z)}{X(z)} = H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 - \sum_{k=1}^{N} a_k z^{-k}}. \tag{2.4}$$

where $H(z)$ is the transfer function.

The difference equation is arranged as a transfer function with the form of $Y(z)/X(z)$. The roots of the numerator and denominator polynomials are referred to as *poles* and *zeros*. Locations of the poles and zeros in a transfer function will directly affect the response of the filter. The pole and zero locations are determined by the values of the coefficients.

When a filter requires the use of higher order polynomials, then multiple second-order sections can be cascaded. If the order of the numerator or denominator in the transfer function is not an even power of two, certain coefficients can take on the value of zero to gain the desired order.

To illustrate the utility of the z-transform, consider the following difference equation with corresponding transfer function $H_1(z)$:

$$\begin{aligned} 72y[n-4] - 126y[n-3] + 67y[n-2] - 14y[n-1] + y[n] = \\ -30x[n-4] - 16x[n-3] - 0.5x[n-2] - 5.5x[n-1] + 1x[n]. \end{aligned} \tag{2.5}$$

By rearranging terms, we obtain the following transfer function in terms of second-order sections using the z-transform:

$$H_1(z) = \left[\frac{(1 + 2.5z^{-1} + 3z^{-2})}{(1 - 5z^{-1} + 4z^{-2})}\right]\left[\frac{(1 + 3z^{-1} - 10z^{-2})}{(1 - 9z^{-1} + 18z^{-2})}\right] \tag{2.6}$$

or by using the direct form, the difference equation can be written as

$$\begin{aligned} Y(z) &= X(z) - 5.5X(z)z^{-1} - 0.5X(z)z^{-2} - 16X(z)z^{-3} - 30X(z)^{-4} \\ &\quad + 14Y(z)z^{-1} - 67Y(z)z^{-2} + 126Y(z)z^{-3} - 72Y(z)z^{-4}. \end{aligned} \tag{2.7}$$

The structure shown in Figure 2.1 shows the filter implemented in the direct form and Figure 2.2 shows the canonic form in second-order cascade sections.

The direct form is the direct realization of a filter taken from the difference equation. The difference equation uses the input $x[n]$ and output $y[n]$. The system transfer function can be found from the difference equation. The direct form is known as Direct Form I, is shown in Eq. 2.6. Figure 2.1 shows how Eq. 2.6 is implemented. The canonic form results when the minimum number of delay elements are used to realize the filter. The canonic form is known as Direct Form II, is shown in Eq. 2.8. Figure 2.2 shows how Eq. 2.8 is implemented. The two structures implement the same poles and the same zeros.

2.2.3 Quantization Errors for Digital Filters. Some of the errors that occur within a digital filter are due to quantization. These quantization errors affect the performance of the digital filter. Quantization errors are classified into three categories:

1. analog-to-digital conversion noise,
2. coefficient quantization, and
3. roundoff noise[16].

The direct form realization of the filter follows directly from the difference equation. However, different schemes can be used to implement the same filter design, but the different realizations will have a different number of multipliers, adders, and memory elements. However, for infinite impulse response structures, the roundoff noise increases dramatically. The cause of this increase in sensitivity is due to the poles in the transfer function [10:chapter 7]. The direct form of realization for FIR filters will be less sensitive when compared to the IIR structure, due to the non-recursive structure.

2.2.3.1 Quantization in Analog-to-Digital (A-D) Conversion. Quantization errors are introduced by the A-D converter. This process transforms a continuous-time signal into a sampled discrete-time signal. The continuous-time signal is comprised of a continuum

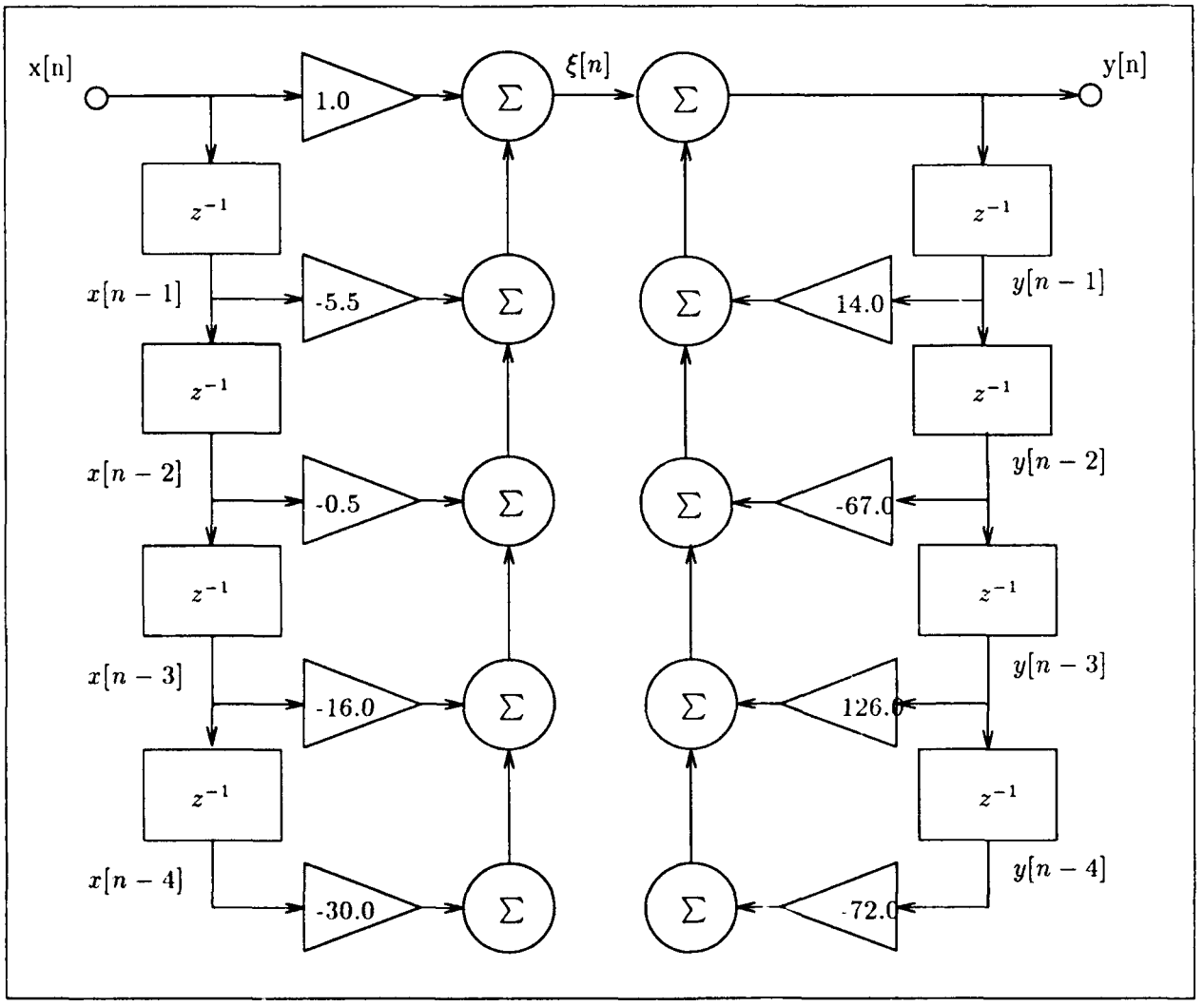


Figure 2.1. Direct Form 1 IIR digital filter.

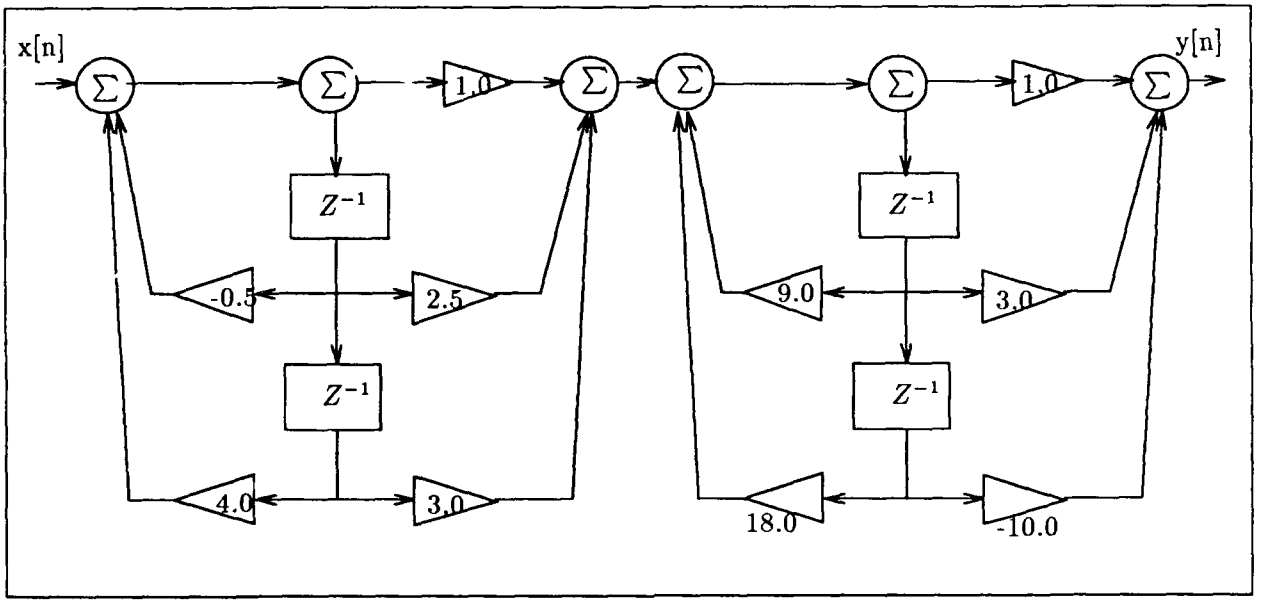


Figure 2.2. Second order sections cascaded in Direct Form II.

of points varying in amplitude. A digital signal can only have a finite amount of time between sampled points and a finite number of discrete levels to comprise the signal. There are finite levels available for an A-D converter. For example, an 8-bit A-D converter has only 256 discrete levels to represent a continuous signal. The difference between the actual signal and the quantized version of the signal constitutes quantization error.

2.2.3.2 Coefficient Quantization. Quantization of the filter coefficients directly affects the system transfer function. In general, designs for filters depend upon irrational numbers. The infinite-length binary word representation does not exist in hardware. The rounding off of the irrational numbers used as weights for the filter coefficients changes the filter's transfer function.

Since there exists a limit on the word register length, not all decimal numbers will map exactly into binary. Numbers that are an integer multiple of 2^{-B} bits, where B is the number of bits to right of the binary point, will map exactly; all others will require quantizing.

2.2.3.3 Quantization in Roundoff Noise. When a multiply is performed on two words in binary format, the result will require twice the number of bits of the starting binary

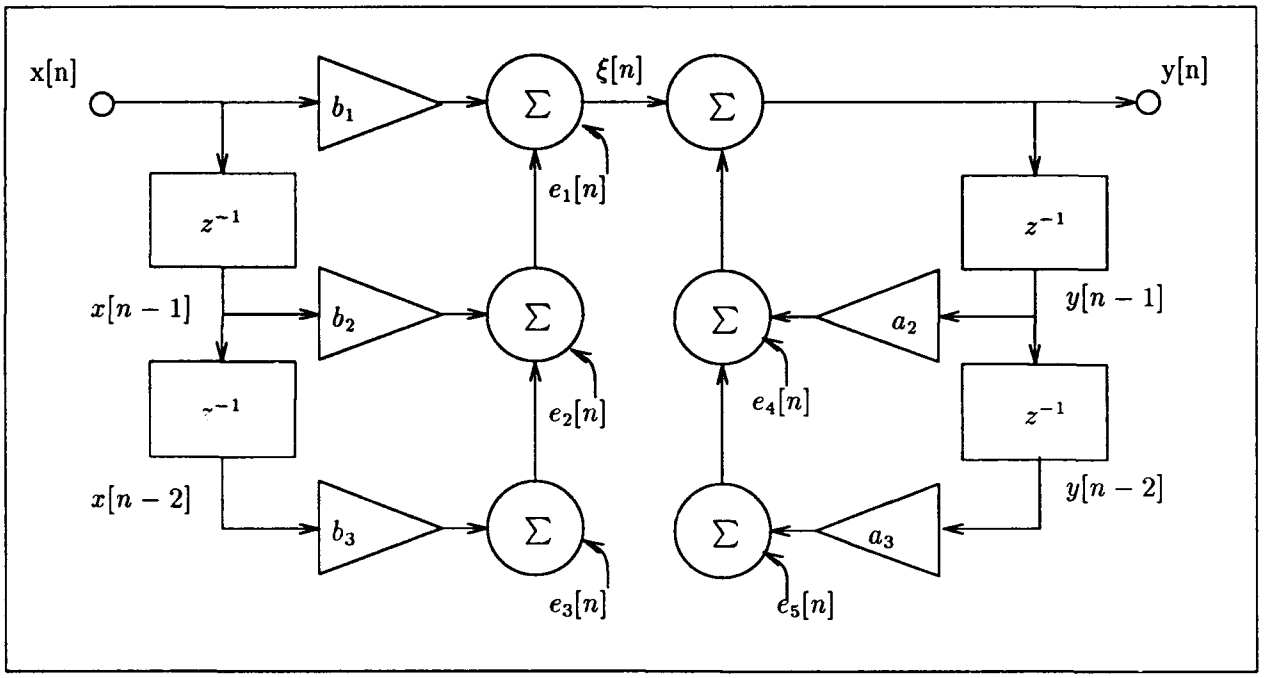


Figure 2.3. Linear noise modeling for direct form I showing the injection of noise sources after each multiplication.

format. Since the structure of memory organization is fixed by the digital hardware, the least significant bits in the binary format are truncated or rounded. In either case, this type of error is called roundoff noise.

The analysis of roundoff assumes that roundoff noise can be modeled as a zero-mean white noise source with a uniform probability distribution function [8:188]. The variance of this noise source is then $X^2/12$, where X is the quantization step size in the binary word. The noise sample can take on a value between the interval $(-X/2, X/2)$. Each point in the filter where multiply operations occur can be modeled as having the roundoff noise added to the result. These noise sources are uncorrelated with other noise sources and the sampled signal. Figures 2.3 and 2.4 show the modeling of roundoff noise by the noise sources added into the filter structure for the direct form and the cascade form.

The form of realization determines the extent of change in the output of a filter due to roundoff noise. Computation of the effects of roundoff noise involves the addition of a noise source where rounding occurs. The added noise source would then directly sum with

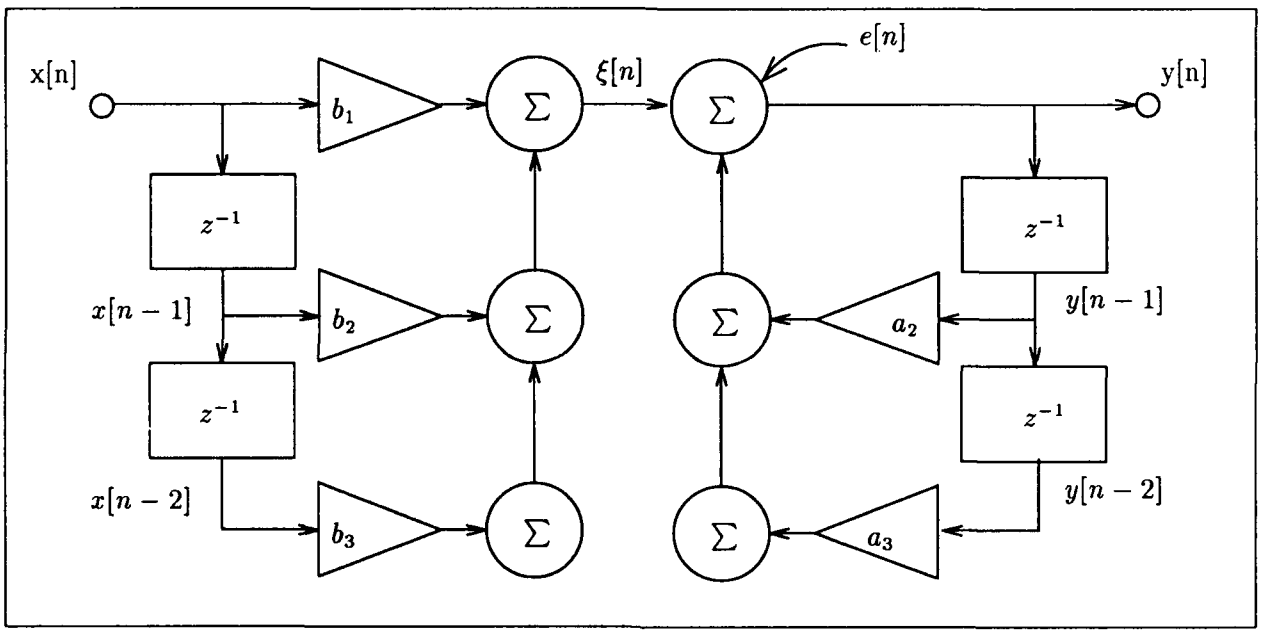


Figure 2.4. Linear noise modeling for direct form I showing superposition of noise sources into one noise source injected into the system.

the output at that particular node. By the use of superposition of the noise sources, all the injected noise sources can be represented by one noise source.

The feedback coefficients will filter the roundoff noise generated within the filter. However, the feedfoward terms will not filter the roundoff noise generated. The feedfoward terms only contribute to the roundoff noise. Figure 2.3 shows the paths for the feedfoward and feedback coefficients. The output power from the noise source can be found by using simulation or by analytical techniques. This is further discussed in Section 3.5.2.

2.2.3.4 Analysis of Filter Response Errors. For the case of FIR systems, the concern of coefficient quantization centers on the zeros in the transfer function. To further illustrate consider,

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n} \tag{2.8}$$

where $h[n]$ is the impulse response of the system [15:345].

When the realization of the filter is done, the coefficients will be quantized to some new value dependent on how accurate the computer can represent coefficients. The impulse response to include the error, can be written as

$$\hat{h}[n] = h[n] + \triangle h[n]. \tag{2.9}$$

By using Eq. 2.8 we can then write the system transfer function as

$$\hat{H}(z) = \sum_{n=0}^{M} \hat{h}[n]z^{-n} = H(z) + \triangle H(z) \tag{2.10}$$

where the amount of change in the the transfer function due to the quantization is found as

$$\triangle\, H(z) = \sum_{n=0}^{M} \triangle h[n]z^{-n}. \tag{2.11}$$

The amount of change in the system transfer function is a linear function of the quantization errors in the difference equation. This output due to coefficient quantization is then added to the output due to the ideal transfer function without the effects of quantization. The new filter function is the sum of the ideal transfer function added to the transfer function due to coefficient quantization. The scheme is shown in Figure 2.5.

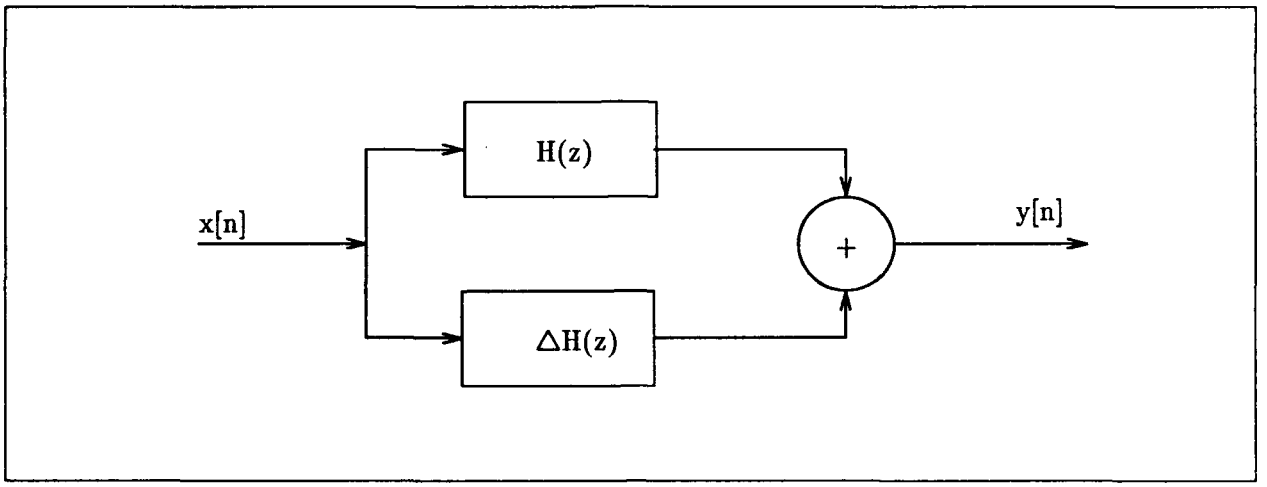


Figure 2.5. Model to calculate the output of a transfer function to include the effects due to coefficient quantization shown as $\triangle H(z)$.

2.3 Roundoff Noise in Digital Filters

Because roundoff noise effects the performance of the digital filter, the designer is faced with how to minimize this quantity. Lee attempts to demonstrate how to minimize the roundoff error using fixed-point arithmetic and sinusoidal inputs to the filter designs [13:424]. Each of the roundoff noise sources are modeled as white noise with uniform pdfs and are independent from other noise sources. Lee introduces a method to help guide the design process called the minimax noise principle. The minmax noise principle provides a means to reduce the effects of roundoff.

2.3.1 Modeling Roundoff Error In Cascaded Filter Sections. Signal overflow can occur within filter structures and adversely affect the assumptions used to find the output power due to roundoff noise [13:422][15:359]. To prevent overflow, the filter gains are ≤ 1.0 at branch nodes where the signal enters. This assumption allows the modeling of roundoff noise to be done, without concern for random overflow conditions.

Jackson modeled roundoff noise as additive white noise sources that are uncorrelated from sample to sample. Jackson also showed the best performance to minimize roundoff noise results from FIR and IIR filters limited to a second order [6:119]. Higher order filters are generated by cascading the second-order sections. Since cascaded sections are placed in

series, another variable introduced to the filter structure is the gain of the roundoff error through each second-order section.

Since each section of the filter will have two poles and two zeroes in the z plane, cascaded sections will have $P!$ ways to order the pairs of zeros and poles for each section, where P is the number of cascaded sections [12:24]. The number of ways to arrange the second-order sections is raised to the second power since there are two singularities per section. The result is the number of ways to order the poles or zeros in a cascaded filter design. The ordering of each second-order section changes the ordering of the gain terms. The section with the highest gain may be last in the cascaded structure or it may come first. The best SNR due to roundoff will be dependent on the order of each section's gain term.

The error in the frequency response can be determined from the effects of roundoff. The mean-square error described has a zero mean and non-zero variance given by the uniform probability distribution [15:353]. Each of the noise sources are effectively summed at the output of the feed forward section. The feedback section will continue to filter the roundoff power signal. The value of the summed noise sources can then be used as an input to the filter's feedback section. This input is then multiplied by the transfer function of the filter's feedback section. Hence, only the poles of the transfer function will recursively filter the roundoff power.

2.3.1.1 The Minimax Approach. This thesis does not use the minimax approach directly. The designer can apply the principle of minimax to manipulate the second-order cascade sections to reduce the roundoff noise term. The total noise output power through the filter is proportional to the ratio between the minimum discrete signal levels and the maximum discrete signal levels throughout the cascaded filter, hence the name minimax.

Lee's paper resulted in a theoretical analysis of the spread of the noise power in terms of the variance. The best design resulted by pairing up the poles and zeros that minimized the dynamic range at the output of the summer nodes in all sections of the cascaded filter. A search is performed on all the pole/zero pairs to pick the best matched pairs to accomplish the minimization ratios. The filter is then built by ordering the cascade sections with the optimized matching pole/zero coefficients for each summer node. The use of the minimax approach improves the signal to noise ratio of the designed filter, thus minimizing the roundoff noise [13].

2.4 Summary

Two forms of finite precision effects exist when implementing a digital filter. The use of the z-transform is introduced to model filter performance. Two types of filter structures are used as a baseline to model the digital filters: the Direct Form I, and the Direct Form II. Implementation of a difference equation using Direct Form I and Direct Form II is given. The representation of coefficients limited in resolution due to the number of bits used will affect the transfer function. Theory to represent the coefficient quantization is persented. Roundoff noise occurs after multiplication due to the result being limited to the length of the multiplicants.

III. Theory of Finite-Precision Effects

3.1 Overview

This chapter presents specific areas of theory to explain the methods used with this digital filter analyzer software tool. The use of z-transforms [15:866] are used as the means to identify the filter's response to input signals. This thesis investigates two types of structures and realizations of digital filters. The first is the direct form that follows directly from the difference equation. The second form is cascade realization. This form requires effort on the part of the designer to manipulate the difference equation to correctly attain factored sub-sections to form the realization. The cascade realization requires the use of poles and zeros that are complex conjugates in order to build the second-order sections.

After the coefficients for the digital filter are determined, the particular implementation chosen can adversely affect filter performance. The amount of degradation can be such that the theoretical infinite-precision performance specifications are not met by the actual finite-precision system. The problems associated with quantization and roundoff are always associated with the use of finite wordlength registers (the cause of finite-precision arithmetic). This thesis provides the designer a tool to immediately see the results of the specific implementation and wordlength limitations in terms of the realized $H(z)$. This chapter shows the effects of coefficient quantization and roundoff noise generation in digital filters.

3.2 Realization of Difference Equations

3.2.1 Filter Forms Digital filter design requires more than producing a difference equation. Filter design requires the incorporation of the effects of finite-precision arithmetic. This ensures that performance of a system doesn't degrade beyond acceptable tolerances when implemented with finite wordlength registers.

The most common forms to mathematically represent a digital filter are difference equations, unit impulse responses, and z-transforms. From these mathematical forms, two basic structures to implement the filter can be derived. Direct Form I is determined directly from the difference equation [15:296-297]. A variation in the implementation of Direct Form I that has reduced memory storage requirements and computations is called Direct Form II. Direct Form II is also called Canonic Direct Form since this form uses the minimum number of delay elements [15:296]. These two structures are used in this thesis. These basic structures

can include an all-pole design, an all-zero design, or both poles and zeros in the design. The all-zero filter is of special interest since its structure has the quality of linear phase for symmetric taps.

3.2.2 Arranging the Difference Equation. To represent a difference equation using block diagrams we consider a second-order difference equation

$$y[n] = a_1 y[n-1] + a_2 y[n-2] + b_1 x[n]. \tag{3.1}$$

The difference equation reveals a recursive structure and hence an infinitely long impulse response. Since the impulse response is infinite in length, a discrete convolution is impossible to implement. The difference equation written in the z-transform domain becomes

$$Y(Z) = a_1 z^{-1} Y(Z) + a_2 Y(Z) z^{-2} + b_1 X(Z). \tag{3.2}$$

Since all difference equations can be implemented using the transfer function, we write

$$H(z) = \frac{b_1}{1 - a_1 z^{-1} - a_2 z^{-2}}. \tag{3.3}$$

The use of the block diagram in Figure 3.1 shows the direct form implementation and allows us to visualize the computer algorithm.

The direct form is only one method of implementing this filter. There exists many structures for implementing a particular system [2:109-118]; some structures will require fewer computations, be more insensitive to finite-precision effects, and provide increased robustness. The type of structures with decreased computational intensity and increased robustness are most desired. These issues are explored with the tool developed in this thesis.

The implementation of this filter using discrete components becomes a straight forward process. Registers will hold values in the summation nodes, values at the output node, and values from delayed versions of the output. Adders will sum the signals at the summation nodes as well as form the building blocks to perform the multiplication. We note a direct correlation of the number of items in the block diagram to the number of discrete components to actually implement the filter. As we continue, we'll see that the use of alternate forms will have some advantages.

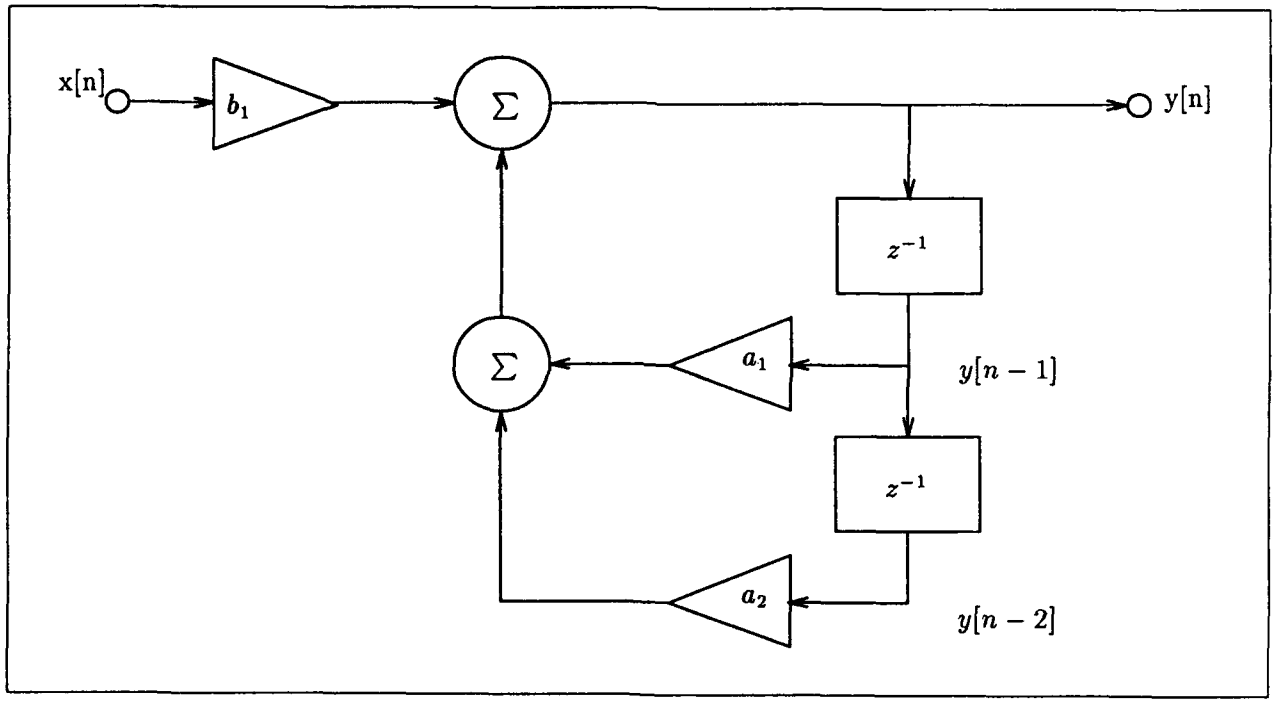


Figure 3.1. Example block diagram for a second-order difference equation.

3.2.3 Direct Form I. Equation 3.1 can be written to include many more coefficients as in

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]. \tag{3.4}$$

This equation can be thought of as two difference equations. We can re-write Eq. 3.4 as two distinct equations as the following:

$$\xi[n] = \sum_{k=0}^{M} b_k x[n-k] \tag{3.5}$$

$$y[n] = \sum_{k=1}^{N} a_k y[n-k] + \xi[n]. \tag{3.6}$$

This set of difference equations shows the output $y[n]$ is found with an additional sequence $\xi[n]$. The value of $\xi[n]$ must be computed first. Then the value of $\xi[n]$ can be used by the second difference equation. From an implementation issue, this adds more clock cycles to control when $\xi[n]$ can be used. The block diagram in Figure 3.2 is from direct application of Eq. 3.6.

By observation of Figure 3.2 one can visualize the two sections in the block diagram. The sections are separated by the term $\xi[n]$ that serves as an output from the first section and as an input to the second section.

Since the system output from two frequency transfer functions in serial is independent of the order of the two transfer functions, we can make the following observations. Two sequences are produced in Figure 3.2. The first is the intermediate result, $\xi[n]$, that is produced by the input $x[n]$. The second result is the output sequence value $y[n]$ term. This constitutes two independent transfer functions. Rearrangement of the transfer functions will not cause any change in the overall system performance.

3.2.4 Direct Form II. The transfer function

$$H(z) = H_1(z)H_2(z) = H_2(z)H_1(z) = \left(\frac{1}{1 - \sum_{k=1}^{N} a_k z^{-k}}\right)\left(\sum_{k=0}^{M} b_k z^{-k}\right) \tag{3.7}$$

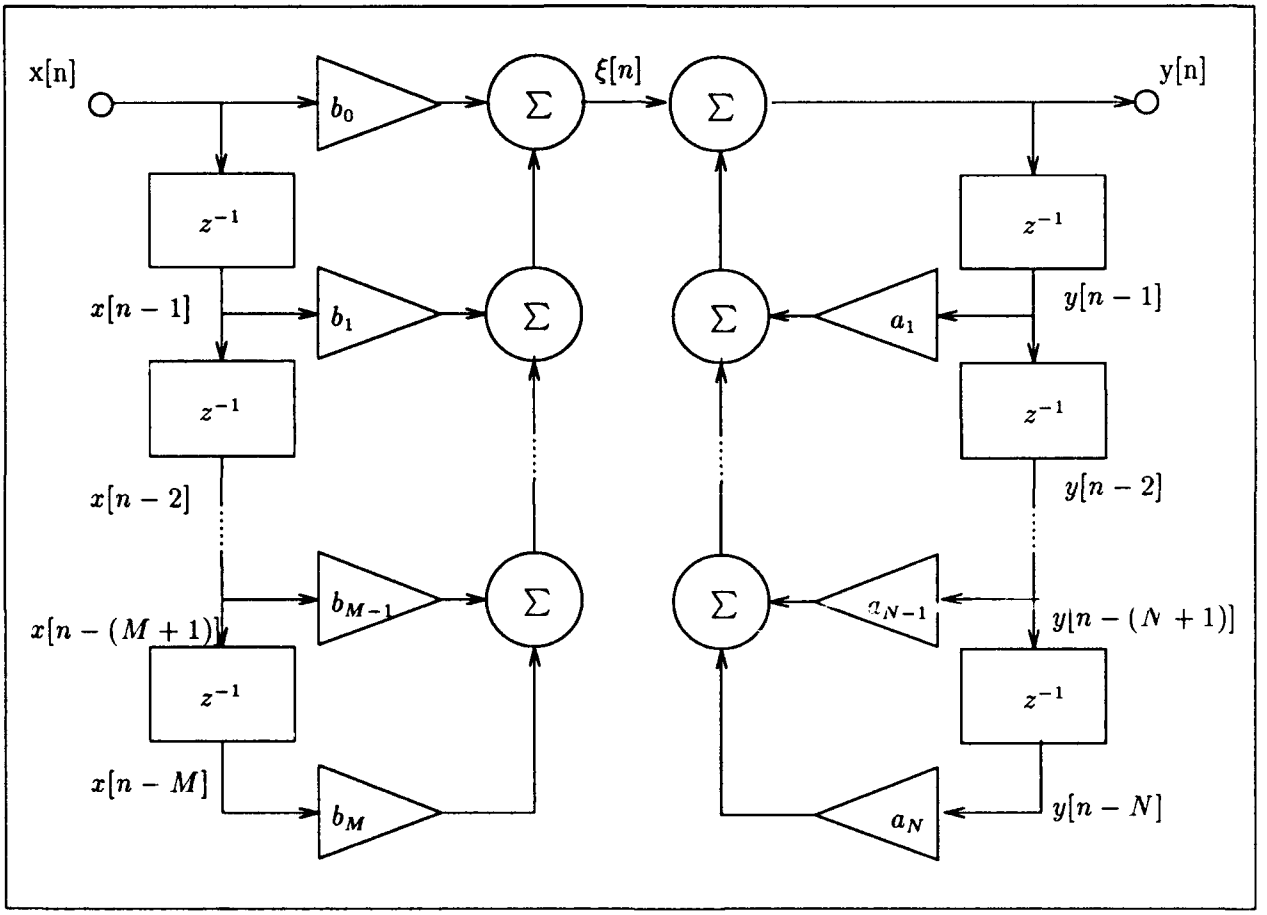


Figure 3.2. Block diagram for the general order difference equation in direct form I.

where

$$H_1(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right)$$
$$H_2(z) = \left(\frac{1}{1-\sum_{k=1}^{N} a_k z^{-k}}\right)$$

shows how the transfer functions can be interchanged. By setting the intermediate value to $\xi[n]$, the following is found from Figure 3.2

$$Z\{\xi(z)\} = H_1(z)X(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) X(z) \tag{3.8}$$
$$Y(z) = H_2(z)Z\{\xi(z)\} = \left(\frac{1}{1-\sum_{k=1}^{N} a_k z^{-k}}\right) Z\{\xi(z)\}. \tag{3.9}$$

Then from Figure 3.3 we can write the following where $P(z)$ is an intermediate result as:

$$P(z) = H_2(z)X(z) = \left(\frac{1}{1-\sum_{k=1}^{n} a_k z^{-k}}\right) X(z) \tag{3.10}$$
$$Y(z) = H_1(z)P(z) = \left(\sum_{k=0}^{m} b_k z^{-k}\right) P(z). \tag{3.11}$$

The order of implementation will yield intermediate results that will be different. In the first case as in Eq. 3.8 the intermediate results are dependent on the transfer function $H_1(z)$ that contains the zeros of the overall transfer function. In the second equation as in Eq. 3.10 the overall transfer function is found by the multiplication of the intermediate result with the transfer function containing the poles. The rearrangement of the order of implementation as in Equations 3.10 and 3.11, will still yield the same overall transfer function response.

Since there are two transfer functions in the z-domain in Figure 3.2, the order of each respective transfer function can be changed. The first transfer function can go where the second transfer function is and vica versa. When a design has many cascade sections, the number of ways to arrange the sections is found by taking the factorial of the number of sections [12:24]. With each rearrangement of this form comes a different way to compute algorithmically the filter's response.

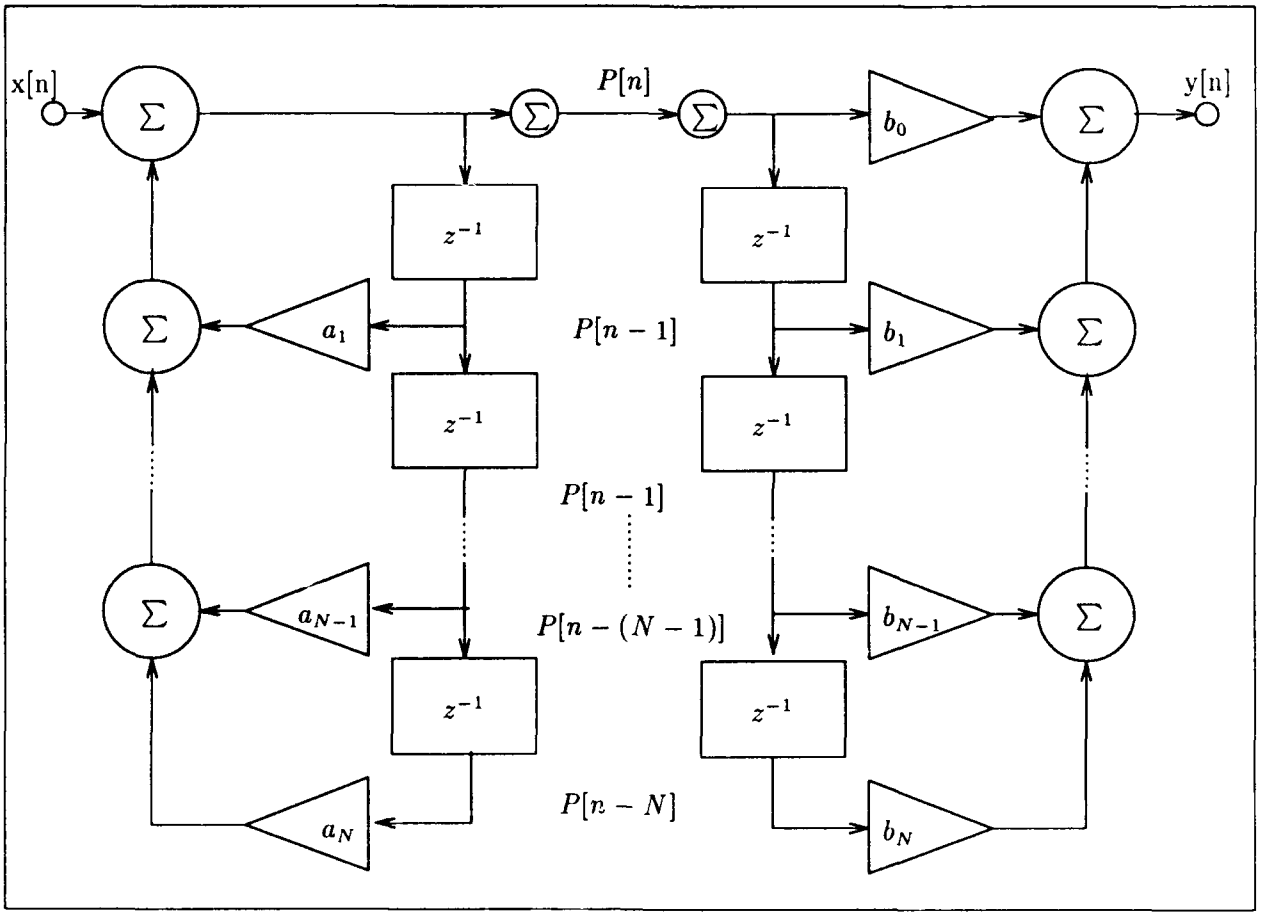


Figure 3.3. Block diagram for the general order difference equation in direct form II.

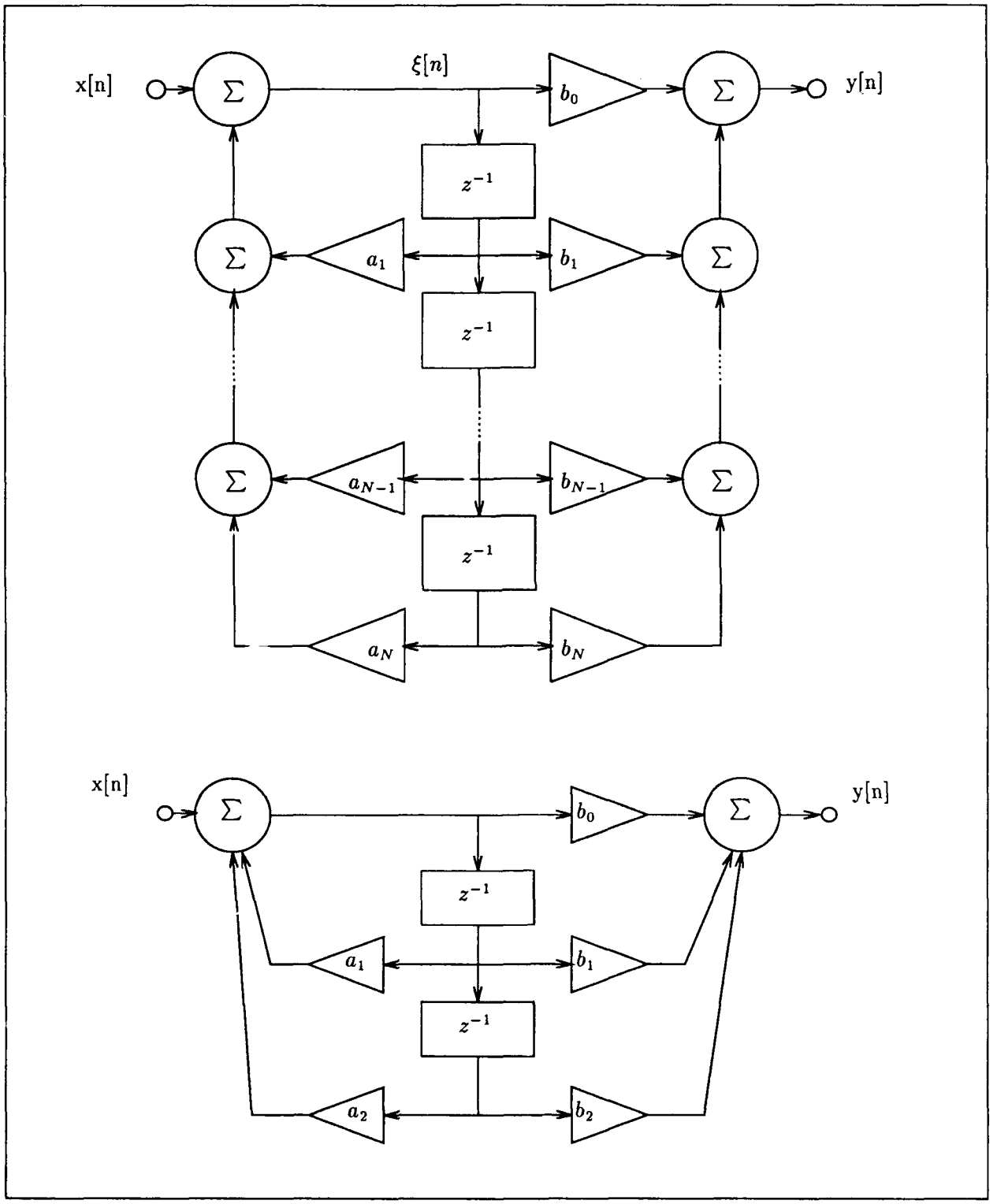


Figure 3.4. Reducing the delay elements to a minimum for the canonic form and the second-order canonic form.

As shown in Figure 3.2 and Figure 3.3, the total number of delay elements is (N + M). Figure 3.3 shows that the intermediate value P[n] and its delayed versions traverse both sets of delay elements. This will result in the same value of P[n] in both legs. Therefore, the entire figure can be re-drawn. This will have the effect of reducing the number of delay elements to a minimum form. The implementation with the minimum number of delay elements is referred to as the canonical form or as the direct form II realization [15:297]. This reduction of delay elements will correspondingly reduce the number of registers needed to implement the filter. The canonic form of a general IIR filter is found in Figure 3.4. Also included is the canonic form for a second-order section in Figure 3.4. This second-order canonic form is the basis for much of the research done in this thesis. The use of second-order sections to build a digital filter has advantages that are clearly seen in the results of this thesis effort.

3.2.5 Cascading Second Order Direct Form II Sections. In cascade design, the output of one section is connected to the input of the next section. This is repeated until the final output is determined. Two linear time invariant systems will behave exactly alike when the impulse responses are convolved to form one impulse response as shown in Figure 3.5. The impulse response of a cascade network is independent of the order in which the sub-systems are cascaded.

3.3 Finite-Precision Considerations for the Digital Filter

When the designer implements a digital filter, the first area of concern is the amount of computational accuracy that will be available as well as the number of registers and adders required. The delay elements in a filter all require input, output, and temporary storage of intermediate results. The coefficients used by the filter require their own storage. The basic elements for a discrete-time system include adders, multipliers, delay registers, and memory for storage. All these elements are limited in the resolution of numerical representation.

3.3.1 Finite-Precision Representation. Every theoretical filter has a unique advantage over hardware filters; theoretical filters avoid finite-precision effects. In hardware, binary values may be limited to only a few bits or include as many as 23-bits or more. The number of bits available for each binary value will effect filter performance, depending on the structure. In general, using fewer bits to represent coefficient values results in a degradation of system performance.

Three basic areas are involved in finite-precision.

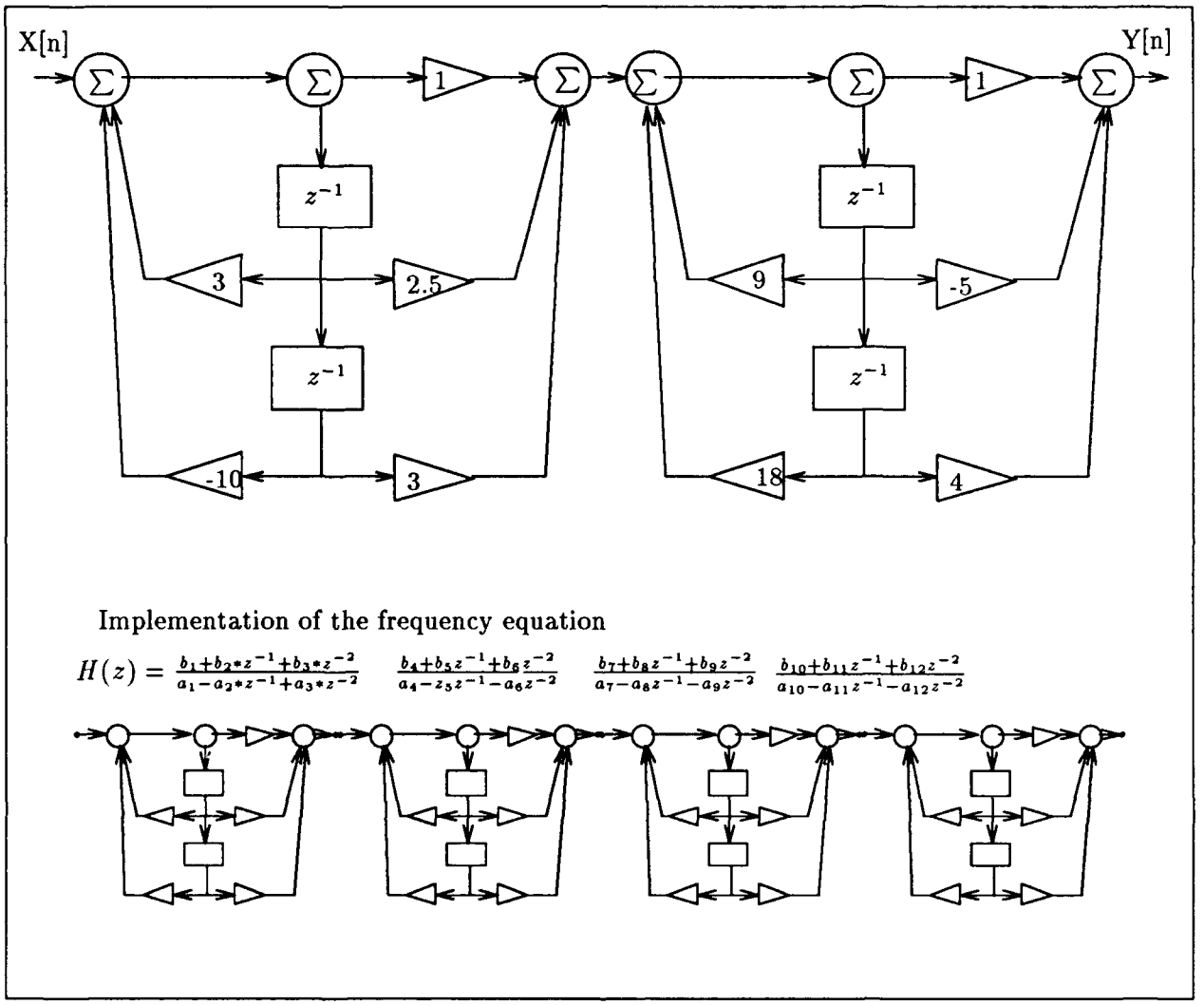


Figure 3.5. Fourth-order filter and an eighth-order filter realized with cascaded second-order sections.

1. Conversion of analog signals to digital representation of those signals results in quantization errors.
2. Filter coefficient values are rounded to the available resolution dependent on the finite wordlength and the number representation scheme.
3. All intermediate results are rounded within the filter structure due to the finite register wordlength and the number representation scheme.

3.3.2 Floating-Point Notation Used by the Simulator Tool. All internal data values and coefficients are in binary format. The computer that runs the software simulator uses the IEEE single precision standard floating point format. The IEEE floating point format is 32-bits in length. It has 24-bits for the mantissa that includes sign and 12-bits for the exponent to include the sign. That leaves 23-bits to represent a number. The format assumes a one to the left of the binary point and 23-bits available to the right of the binary point. The accuracy of the format then becomes

$$2^{-23} \cong 1.2\text{E}10^{-7}.$$

That means about seven digits are available to represent all numbers.

3.3.3 Fixed-Point Notation Used By The Digital Filter Simulator Tool. The software simulator assumes the use of a format called two's complement, a fixed-point representation. The two's complement register has a bit reserved for the sign of the number. Therefore, (B+1) bits are used to indicate the resolution, where B is the number of bits available to represent the number. Figure 3.6 shows the format for the representation of a two's complement word. All the numbers represented within the filter will be truncated to a maximum magnitude of $(1 - 2^{-B})$. The software simulator has routines to change the single-precision values into the fixed-point notation in order to correctly simulate the two's complement binary format.

The *resolution* of the number is determined by the smallest increment available. This is calculated from the least significant bit position. When fixed point notation is used, the smallest increment is a function of the number of bits to the right of the binary point. A four-bit word (one bit is dedicated to the sign of the number) can represent numbers to the nearest 2^{-3} or 0.125.

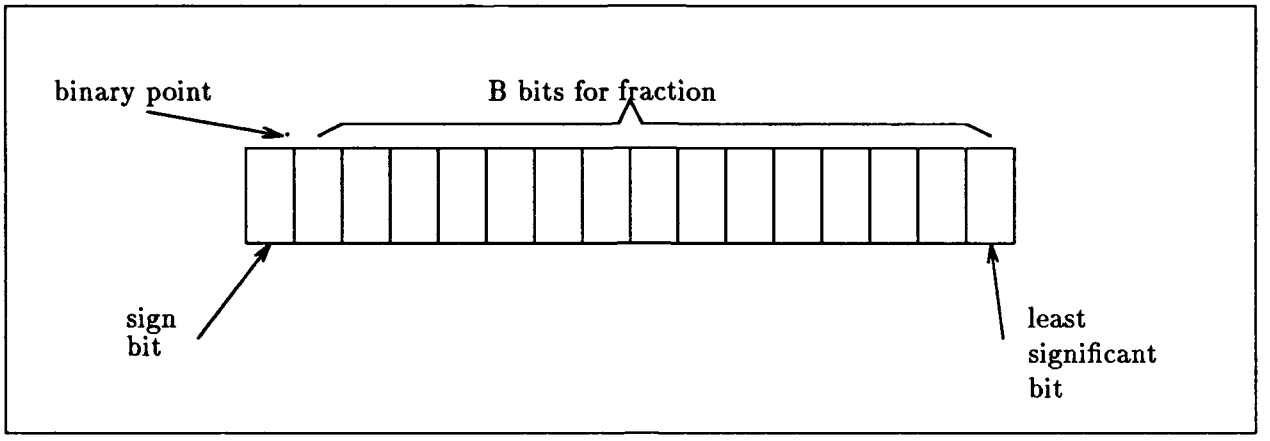


Figure 3.6. Allocation of bits in a fixed-point two's complement representation (16-bit word shown).

Table 3.1 shows an example of two's complement notation using only 2-bits. Table 3.2 shows an example of two's complement notation using 3-bits.

The result of the quantizer routine will allow for a value of $\pm$ unity. This assumes the use of adders with logic circuits that includes a carry bit. The use of a carry bit will expand the accuracy available in the binary representation of the coefficients and intermediate results using two's complement.

Table 3.1. Representation of numbers with 2-bits.

Binary Number	Integer Values(binary point at left)	One's Complement
2-bit number system		
00	0	+0.0
01	1	1.0
10	2	-0.0
11	3	-1.0

3.4 Quantization Error

The implementation of the design process changes the resulting coefficients through the effects of quantization. Errors occur from the amount of quantization from the full precision coefficient to the quantized coefficient. Also, the location in the z-plane of the singularities changes the amount of quantization that occurs. These areas are discussed.

Table 3.2. Representation of numbers with 3-bits.

Binary Number	Integer Values(binary point at left)	One's Complement
3-bit number system		
000	0	0.0
001	1	0.25
010	2	0.50
011	3	0.75
100	4	-0.0
101	5	-0.25
110	6	-0.50
111	7	-0.75

3.4.1 Pole-Zero Locations To gain some insight into the effects of coefficient quantization, an understanding of the pole-zero locations is necessary. The problems of quantization are easily seen by Figure 3.7. The results from the summation nodes as well as the multiplication with the coefficients must fit into a specific set of allowable steps. Figure 3.7 represents a system with four bits. Three bits contribute to the resolution and one bit determines the sign. Once saturation occurs, the corresponding error will continue to increase without bound.

In Figure 3.7, the quantization error is shown on the bottom. The error value is linear across the quantization value. At the limits of the quantization values, the error is shown to either increase or decrease without bound. The quantizer is in saturation when it is at the maximum value that can be represented with the input signal continuing to increase. Likewise, the error is in cut-off when the quantizer is at the minimum value with the input signal continuing to decrease.

Normalizing the maximum coefficient to a value of unity will avoid the saturation regions from multiplications since the coefficients will be numbers less than one. However, it can be seen that a summation node can exceed the limits of even the carry bit. Two's complement can retain the correct value if it unwraps and its use is most common [15:329].

The effect of quantization is to restrict the locations of poles and zeros. If a singularity location doesn't exactly match the possible locus of points on the grid in the z-plane, a shift will be made in the singularity location. The singularity will have to shift to the closest one of four possible grid locations (two in the real axis and two in the imaginary axis). This shifting causesthe transfer function $H(z)$ to be different from the design using infinite

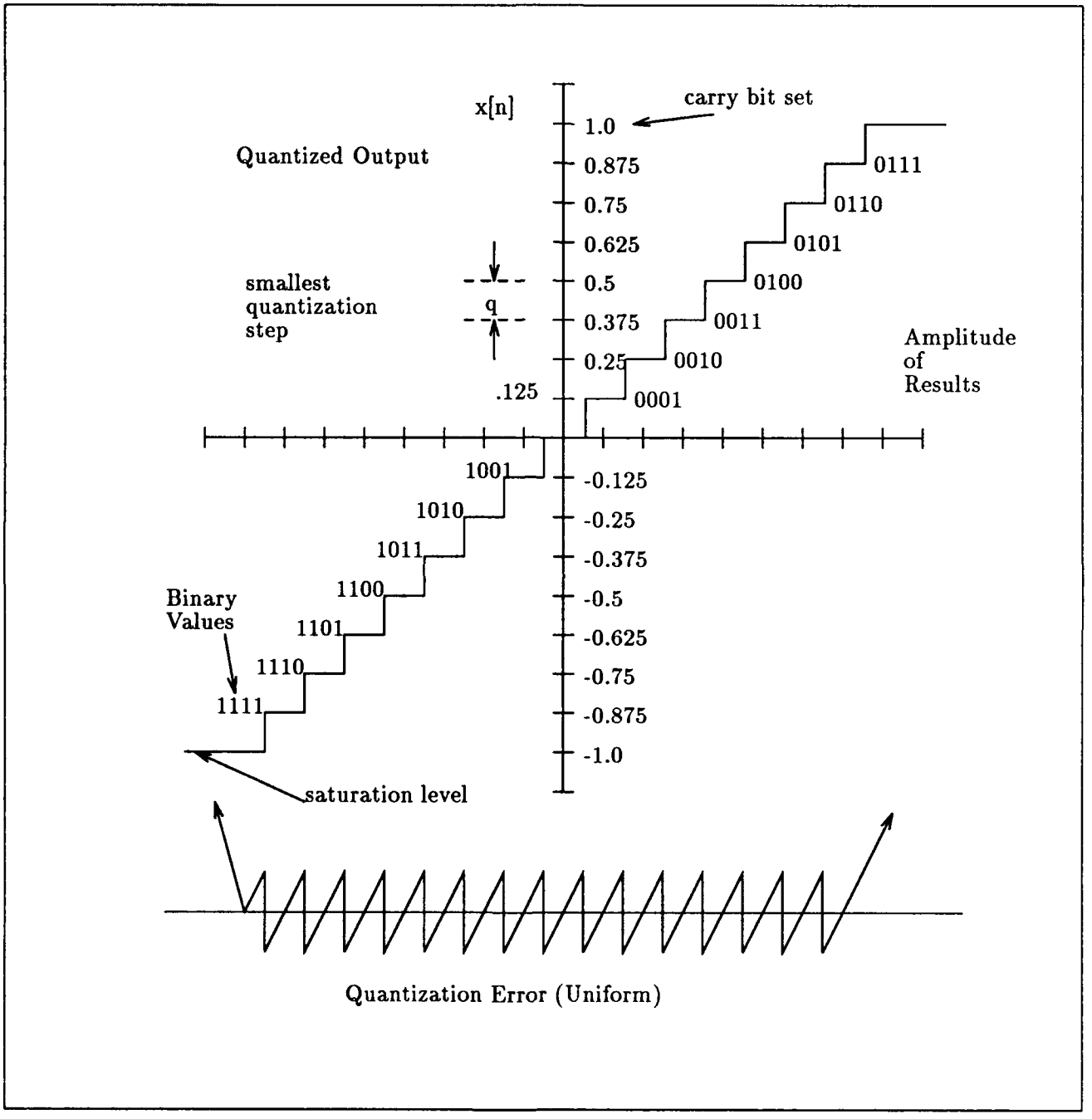


Figure 3.7. Number quantization effects with 4-bits, Two's complement rounding.

precision. In some cases this shift can cause instability. Consider a causal IIR filter in both direct and cascade form given as

$$H_C(z) = \frac{1}{(1-0.901z^{-1})(1-0.943z^{-1})} \quad \text{cascade form} \tag{3.12}$$

$$H_D(z) = \frac{1}{(1-1.844z^{-1}+0.849643z^{-2})} \quad \text{direct form.} \tag{3.13}$$

It can be seen that rounding even to the nearest 2^{-4}, the cascade form transfer function becomes

$$\tilde{H}_C(z) = \frac{1}{(1-0.90z^{-1})(1-0.95z^{-1})} \tag{3.14}$$

with poles at z=0.90 and z=0.95. This filter is stable since the poles are inside the unit circle [15:31]. But consider using the direct form

$$\tilde{H}(z) = \frac{1}{(1-1.85z^{-1}+0.85z^{-2})} = \frac{1}{(1-0.85z^{-1})(1-1.0z^{-1})} \tag{3.15}$$

with poles at z=0.85 and z=1.00. The pole on the unit circle causes this filter to become unstable [11:399-400]. The process of quantization restricts the possible values the singularities can be for both the poles and zeros. Figure 3.9 shows the case for poles only assuming the use of two's complement notation.

3.4.2 Quantization of Singularities. There exists a grid of possible pole-zero locations in the z-plane for a transfer function. Consider the following complex conjugate poles,

$$H(z) = \frac{1}{(1-re^{j\theta}z^{-1})(1-re^{-j\theta}z^{-1})} \tag{3.16}$$

$$= \frac{1}{(1-2rcos(\theta)z^{-1}-r^2)z^{-2}}$$

$$y[n] = x[n] - 2r\cos\theta y[n-1] + r^2 y[n-2]$$

$$Y(z)[1+2r\cos\theta z^{-1}-r^2z^{-2}] = X(z). \tag{3.17}$$

Figure 3.8 shows the direct form implementation of this difference equation in Eq. 3.17. When the coefficients are quantized, the value of r^2z^{-2} can take on seven positive quantized values and zero while the other coefficient, $2rcos(\theta)z^{-1}$ can take on seven positive values, zero,

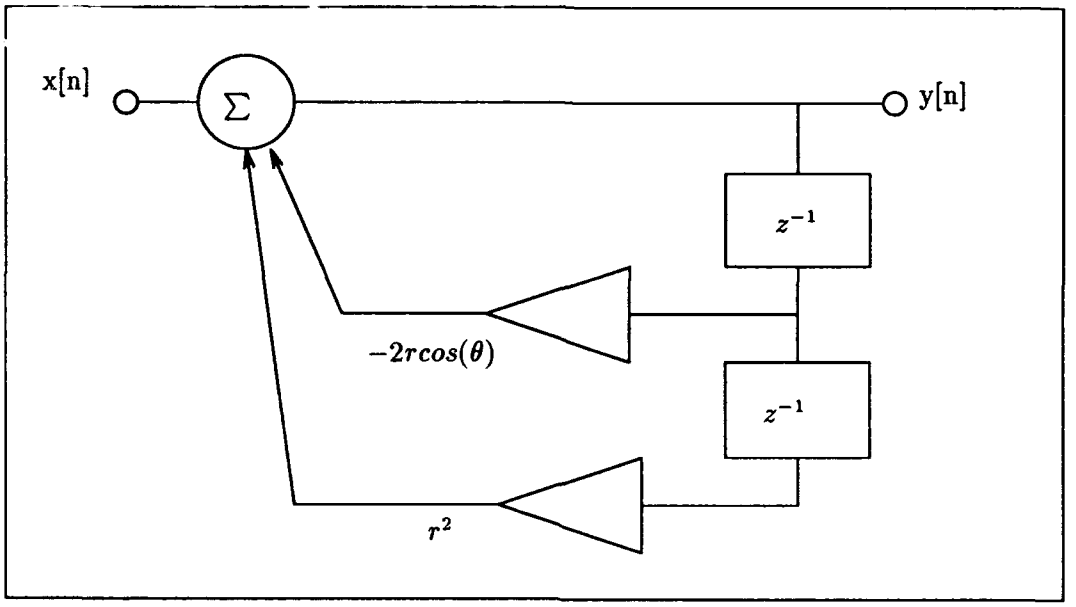


Figure 3.8. Direct form implementation for a complex-conjugate pole pair.

and seven negative values (*eight negative values if two's complement* is used). Coefficient quantization restricts the positions of the poles as shown in Figure 3.9.

Figure 3.9 shows the possible zero singularities for 4-bit word lengths. Only one quadrant is shown. The other quadrants in the z-plane are mirror images of the one shown. Notice in Figure 3.9 that the grid is more sparse around the real axis. If a design has zeros (or poles) that are close to the real axis, they will be shifted further. This will certainly cause more degradation in implementation [15:342].

3.4.3 Sensitivity of Pole/Zero Locations. A designer needs to be aware of the limited set of possible locations in the z-plane for all poles and zeros. If pole and zero singularities are real values, or where the angle from the real axis in the z-plane is close to zero, longer word lengths may be needed to maintain desired filter performance. Narrowband lowpass filters and narrowband highpass filters have poles that fall near $|z| \approx \pm 1.0$ [8:185]. These filters will show higher sensitivity to the coefficient quantization process. Also, numerical analysis [8:183-185] shows that coefficient sensitivity will get worse as the order of the polynomial increases. Higher order polynomials will have a higher sensitivity to the roots. Accuracy must then increase for the coefficients in higher order polynomials to maintain acceptable results.

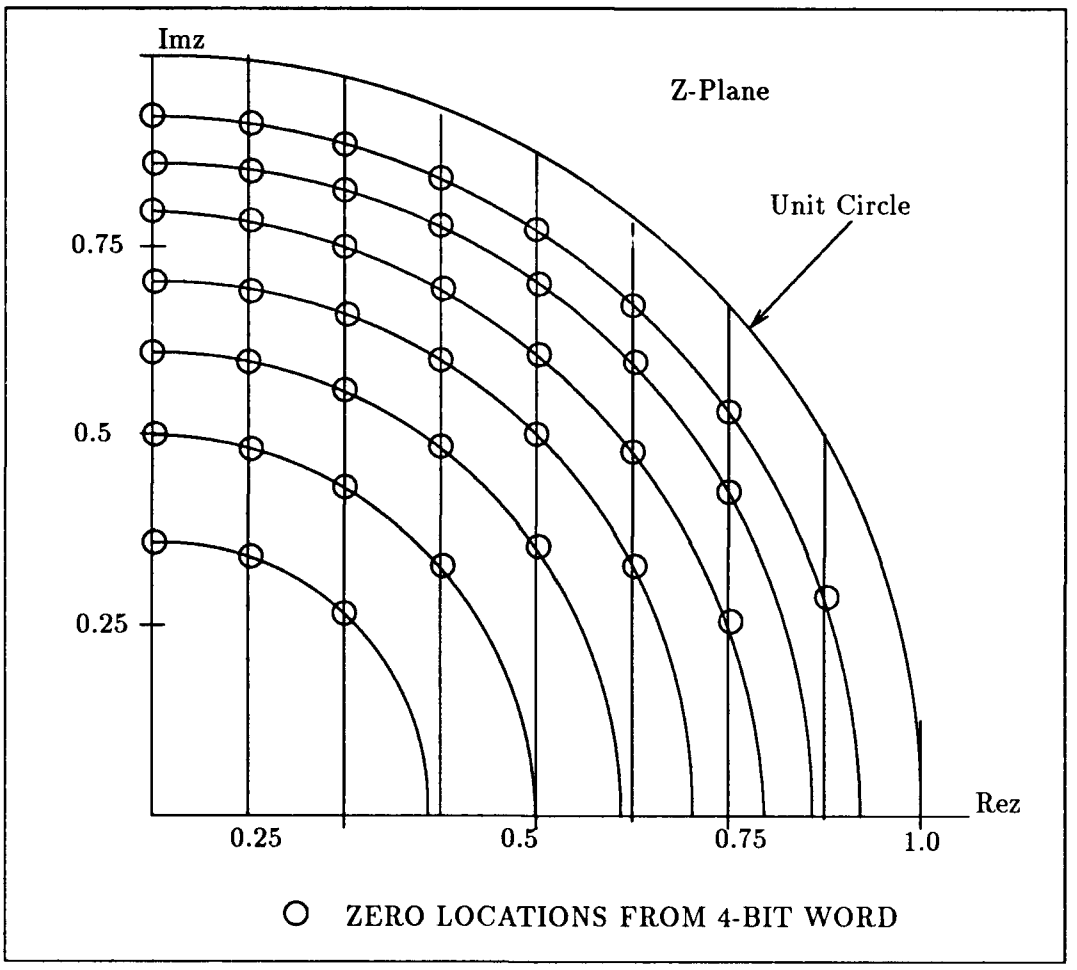


Figure 3.9. Possible locations for singularities of a second-order FIR direct form implementation.

3.5 *Linearized Models For Roundoff Errors*

The translation from analog to digital will result in a quantization error as shown in Figure 3.7 and defined by

$$e[n] = \hat{x}[n] - x[n]$$

where $\hat{x}[n]$ is the quantized sample.

When a three-bit quantizer is used, then the error will always maintain a magnitude within the smallest quantization step 2^{-B} where B is the number of bits. This is shown by the error in quantization at the bottom of Figure 3.7 limited to the size of the quantization step. The error magnitude is linear across the quantization level. The variance is found by [15:838]

$$V(X) = E(X^2) - E^2(X) \tag{3.18}$$

Substituting into Eq. 3.18 with the definition of the expected value and the second moment we have

$$V(X) = \int_{-\infty}^{\infty} x^2 f(x) dx - \left[\int_{-\infty}^{\infty} x f(x) dx\right]^2 \tag{3.19}$$

where $f(x)$ is the probability distribution function over the smallest quantization step. The variance then depends only upon the range of the quantization interval. The variance can then be written as

$$\sigma_e^2 = \frac{q^2}{12} \tag{3.20}$$

where q is the quantization step equal to 2^{-B}.

The roundoff noise can then be modeled by the variance of a uniformly distributed random process. The modeling of the roundoff noise as an addition of a noise source was proposed by Jackson [7]. Each noise source requires the three assumptions:

1. The error sequence from each noise source model is a sample function of a stationary random process.
2. Each error sequence modeled as a noise source is uncorrelated with the input sequence $x[n]$ and uncorrelated with all other noise sources.
3. The probability distribution of the error process is uniform over the smallest quantization step associated with the process.

to help simplify the analysis considerably. These assumptions are not valid when the system under study has a step function for $x[n]$. However, the type of systems most commonly

implemented will have a much more complicated signal for input. The input signal needs to fluctuate in an unpredictable manner in order for these assumptions to hold. Heuristically, these assumptions are valid when the signal is complex and the quantization step small, so the input signal sufficiently traverses a majority of the quantization steps. For further treatment, the reader is directed to studies by Widrow [18].

3.5.1 Signal-to-Quantization Noise By re-writing Eq. 3.20 with the measure for a quantization step we have

$$\sigma_e^2 = \frac{q^2}{12} = \frac{2^{-2B}}{12} \tag{3.21}$$

where $q = 2^{-B}$ that is the quantization interval and σ_e^2 is the variance due to the error in quantization. The number of bits b, can be represented as $(B+1)$ where B is the precision and the one is for sign. The quantization step can be written as

$$q = \frac{2L}{2^{B+1}} = \frac{L}{2^B}$$

where L is the range of the quantizer when the pdf is calculated.

The signal-to-noise ratio can be defined as

$$SNR_Q = 10log_{10}\left(\frac{\sigma_x^2}{\sigma_q^2}\right) \tag{3.22}$$

where σ_x^2 is the signal power and σ_q^2 is the quantization noise power. The signal-to-noise ratio can then be re-written by substituting in for the signal powers as

$$SNR_Q = 10log_{10}\left(\frac{12 \cdot 2^{2B} \cdot \sigma_x^2}{L}\right) \tag{3.23}$$

$$= 6.02B + 10.8 - 20log_{10}\left(\frac{L}{\sigma_x}\right). \tag{3.24}$$

From Eq. 3.24 we find that the signal-to-noise ratio increases by approximately 6dB with each bit added to the word length of the registers. The power in the term σ_x is the rms value of the signal. The range of the quantizer L is fixed. When the input signal power expressed by σ_x is too large, then distortion will result known as saturation. When σ_x is

small, then the last term in Eq. 3.24 will decrease the signal-to-noise ratio. The optimum value for σ_x is where the input to system uses the full range of values possible.

3.5.2 Modeling Of Roundoff Noise. The autocorrelation function is used to model the roundoff noise. To start, consider the autocorrelation output relation R, for development refer to [17:222-238], as

$$R_{YY}(k) = R_{XX}(k) * h(k) * h(-k). \tag{3.25}$$

Equation 3.25 can be re-written by noting that the convolution in the time domain is multiplication in the frequency domain. Also, the fourier transform of the autocorrelation results in the power spectral density. The value of the power spectral density at $\omega = 0$ is the total power in the process. Hence the total power is designated by the use of variance. The magnitude squared of the effective transfer function is equivalent to the convolution of $h(k)$ with $h(-k)$. The following result is found as

$$\frac{\sigma_o^2}{\sigma_i^2} = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega \tag{3.26}$$

where σ_i^2 is the variance of the sum of the input noise sources and σ_o^2 is the variance at the output (the total power).

With the use of Eq. 3.26, the ouput power due to roundoff can be found by modeling the input noise sources. Each noise source is injected just after each multiply as shown in Figure 3.10.

By the use of superposition, all the noise sources are added forming a single noise injection point shown in Figure 3.11, and can be written as

$$e[n] = e_1[n] + e_2[n] + e_3[n] + e_4[n] + e_5[n]. \tag{3.27}$$

The output $\hat{y}[n]$ is the estimated output from the output of the system plus the output due to the injected error in the filter.

Provided the noise sources are independent of the input and independent from all other noise sources, the variance can be found using Eq. 3.21 we have,

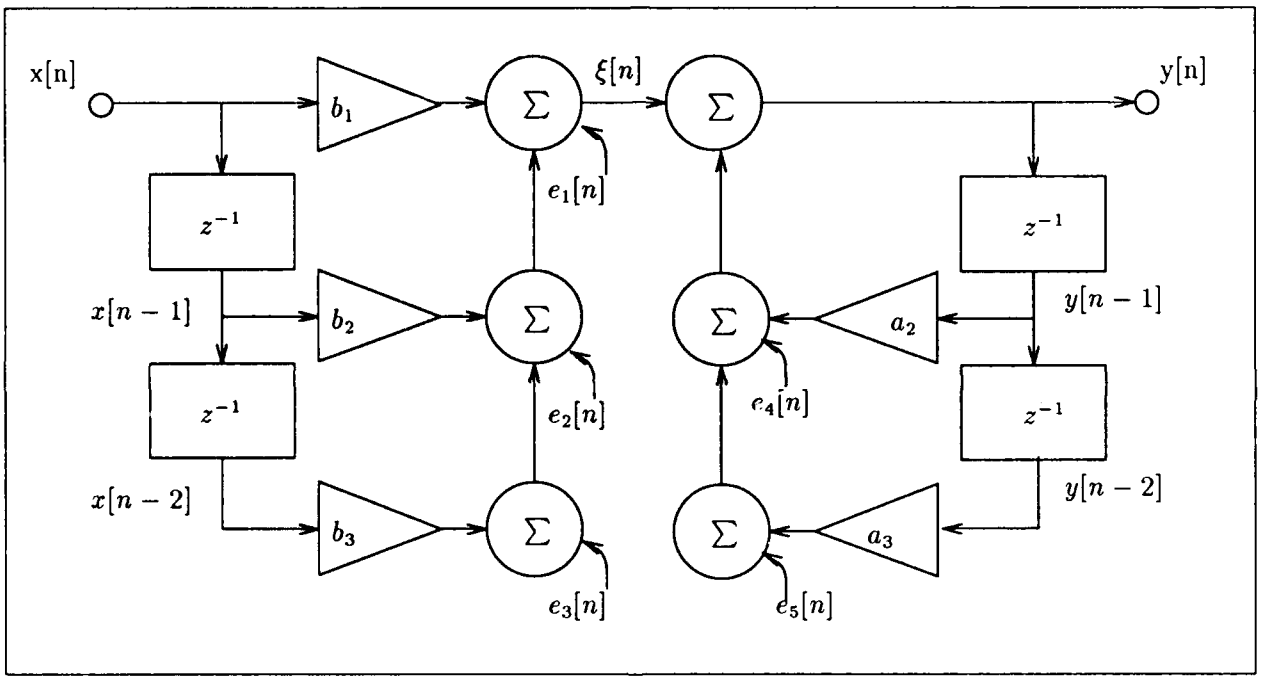


Figure 3.10. Linear noise modeling for direct form I showing the injection of noise sources after each multiplication.

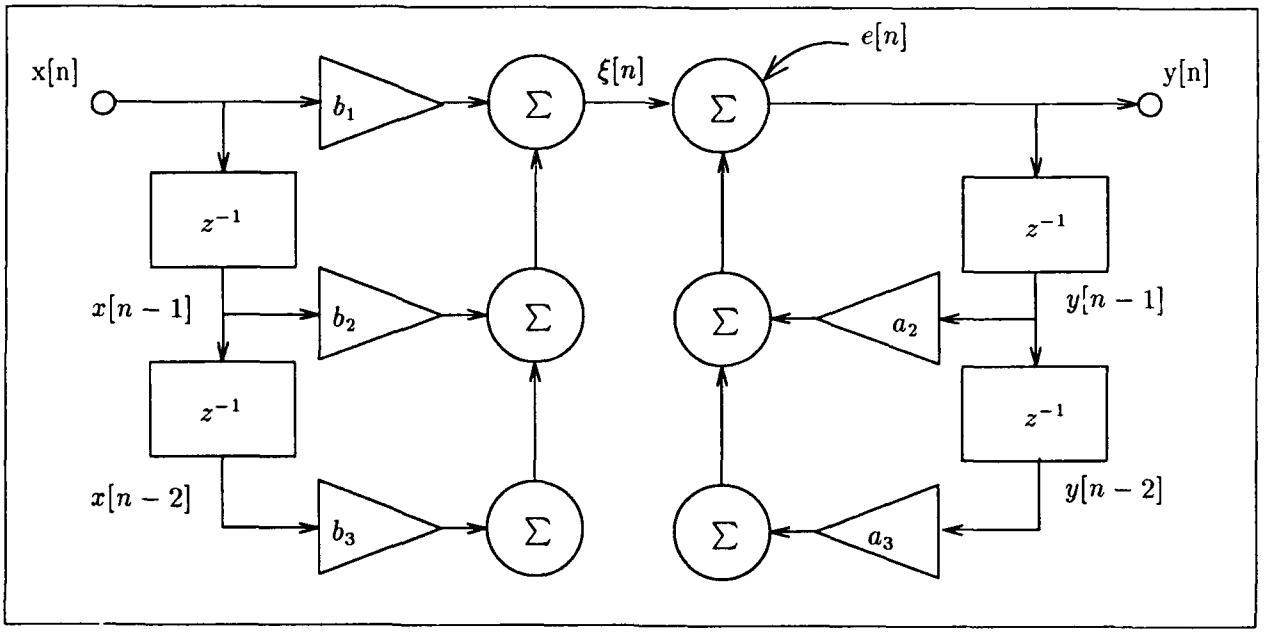


Figure 3.11. Linear noise modeling for direct form I showing the summing of noise sources into one noise source.

$$\sigma_e^2 = \sigma_{e_1}^2 + \sigma_{e_2}^2 + \sigma_{e_3}^2 + \sigma_{e_4}^2 + \sigma_{e_5}^2 \tag{3.28}$$

$$= 5\left(\frac{2^{-2B}}{12}\right) \tag{3.29}$$

for model in Figure 3.11.

This technique can be applied to other direct forms by adjusting for the number of multipliers. To find the variance of the output noise sequence, we integrate the power spectral density. Output power is found by using Eq. 3.25 in terms of power spectral density as

$$S_{yy}(f) = S_{xx}(f)|H(f)|^2 \tag{3.30}$$

$$E\{y^2(t)\} = \int_{-\infty}^{\infty} S_{yy}(f)df \tag{3.31}$$

$$= R_{yy}(\tau)|_{\tau=0}. \tag{3.32}$$

where $E\{y^2(t)\}$ is the total power in the process assuming zero mean.

The output power due to roundoff is found by evaluation of the output autocorrelation function at τ zero. The power spectral density results in the following output noise power equation

$$\sigma_f^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} P_{ff}(\omega)d\omega \tag{3.33}$$

$$= \sigma_e^2\frac{1}{2\pi}\int_{-\pi}^{\pi} |H_{eff}(e^{j\omega})|^2 d\omega. \tag{3.34}$$

where σ_f^2 is the total output noise power due to roundoff, $P_{ff}(\omega)$ is the total power over the frequency range ω, σ_e^2 is the sum of all linear models for roundoff noise injected in the system, H_{eff} is the effective transfer function.

By using Parseval's theorem, we write Eq. 3.34 in terms of the complex variable z as,

$$\sigma_f^2 = \sigma_e^2\frac{1}{2\pi j}\oint_c H_{eff}(z)H_{eff}(z^{-1})z^{-1}dz. \tag{3.35}$$

Equation 3.35 is evaluated using residue theory to integrate over the regions of convergence with n set to the place of origin. Normally, the term n is set equal to zero, since the contour is evaluated at time step [0] for the impulse response from the transfer function.

Using the example shown in Figure 3.11 with the effective transfer function $H_{eff}(z) = B(z)/A(z)$ and by Eq. 3.35, the analysis provides the following solution for the total variance due to the roundoff as

$$\sigma_f^2 = 5\left(\frac{2^{-2B}}{12}\right)\frac{1}{2\pi j}\oint_c H_{eff}(z)H_{eff}(z^{-1})z^{-1}dz \tag{3.36}$$

$$= 5\left(\frac{2^{-2B}}{12}\right)\frac{1}{2\pi j}\oint_c \left(\frac{B(z)}{A(z)}\right)\left(\frac{B(z)}{A(z)}\right)^{-1} z^{-1}dz \tag{3.37}$$

$$= 5\left(\frac{2^{-2B}}{12}\right)\sum_{-\infty}^{\infty}|H_{eff}[n]|^2 \tag{3.38}$$

with

$$|H_{eff}[n]| = \left|\frac{B(z)}{A(z)}\right|.$$

3.5.3 Roundoff Power Computation. This section provides an example of theoretical computation of roundoff power. The use of theoretical computation of the contour integral becomes very tedious and complex for filters beyond fourth order. The results of the theoretical example are presented in Table 3.3.

The theoretical computation is begun by considering the transfer functions

$$F(z) = \frac{1.0}{(z-0.4)} \tag{3.39}$$

$$G(z) = \frac{0.75}{(z-0.5)}\ . \tag{3.40}$$

For purposes of this comparison, the cascade realization is used.

The cascade form is found by the multiplication of each of these two transfer functions as

$$H(z) = \frac{0.75}{(z-0.4)(z-0.5)}. \tag{3.41}$$

Rearranging into a second-order section for implementation yeilds

$$H(z) = \left(\frac{0.0 + 0.0z^{-1} + 0.75z^{-2}}{1 - 0.9z^{-1} - (-0.45)z^{-2}}\right)\ . \tag{3.42}$$

From this transfer function we can write the difference equation as

$$y[n] = 0.75x[n-2] + 0.9y[n-1] - 0.45y[n-2]. \tag{3.43}$$

From Eq. 3.36, the roundoff power is found using the second-order cascade structure as

$$\sigma_f^2 = 3\left(\frac{2^{-2B}}{12}\right)\frac{1}{2\pi j}\oint_c \left(\frac{1.0}{z-0.4}\right)\left(\frac{1.0}{z-0.5}\right)\left(\frac{1.0}{z^{-1}-0.4}\right)\left(\frac{1.0}{z^{-1}-0.5}\right) z^{-1}dz \tag{3.44}$$

noting that only three mulitiplies will result in roundoff power. Simplification yields

$$\sigma_f^2 = 3\left(\frac{2^{-2B}}{12}\right)\frac{1}{2\pi j}\oint_c \frac{z}{(z-0.4)(z-0.5)(2.5-z)(2.0-z)}dz \tag{3.45}$$

The two poles are within the contour of integration at $z = 0.4$ and at $z = 0.5$. The residues are found by

$$\text{residue}_1 = \frac{z}{(z-0.5)(2.5-z)(2.0-z)}\Big|_{z=0.4} = -2.976 \tag{3.46}$$

$$\text{residue}_2 = \frac{z}{(z-0.4)(2.5-z)(2.0-z)}\Big|_{z=0.5} = 3.333 \tag{3.47}$$

Therefore, we now write

$$\sigma_f^2 = 3\left(\frac{2^{-2B}}{12}\right)(-2.976 + 3.333) \tag{3.48}$$

that reduces to the form

$$\sigma_f^2 = 3\left(\frac{2^{-2B}}{12}\right)(0.3573)\text{ , and} \tag{3.49}$$

$$\sigma_f^2 = \left(2^{-2B}\right)(0.0893). \tag{3.50}$$

The use of various wordlengths will provide different values for the roundoff power measurements. For a comparison of different wordlengths, Table 3.3 displays the results. The wordlengths are specified by the number of bits used in the simulation. Table 3.3 shows the results from the theoretical calculations.

Table 3.3. Roundoff power measurements for different wordlengths computed by theoretical means.

Number of Bits	Theory
22-bits	0.0507E-13
20-bits	0.8124E-13
18-bits	0.1300E-11
16-bits	0.2080E-10
14-bits	0.3328E-09
12-bits	0.5324E-08
8-bits	0.1363E-05
6-bits	0.2181E-04

The theory is difficult to work through when the contour integral has repeated roots. The reader can refer to [19:126-127] for further insight into how to evaluate the complex integral.

3.5.3.1 Summary. The two structures for the digital filters used in this thesis were then developed. The two structures, the 1D and the 2D, are used for the digital filter analyzer software tool developed in this thesis. The finite precision effects from the notation were presented. The effects of coefficient quantization were modeled and shown in terms of the placement of the singularities in the z-plane. The means to find the roundoff power were presented along with an example computation.

IV. Digital Filter Analysis Tool

This chapter will cover some the more important subroutines and algorithms used in the software. Methods to simulate the structures of both filter representations are given. The more general sections of code are found in Appendix B, and a User's Manual is found in Appendix C. The user's manual provides a short introduction into the type of filters used and the input file styles. Then a description is given for each of the main menu options.

4.1 Overview of the Digital Filter Analysis Tool

The purpose of this thesis was to build a software tool to study and exemplify the effects of coefficient quantization and calculate roundoff error power generated with the digital filter. These effects occur from the digital representation of numbers within a finite length register. These objectives were to be applied to the two digital structures, direct form and cascade form. The tool developed provides a useful means to study this advanced topic in discrete signal analysis. The software tool is written in FORTRAN. The compiler used was the f77 compiler. I used AFIT's Sun work stations to develop the program.

4.2 The Main Computation Section for Computations

This section of code immediately follows the controlling section where the main menu options are. This computation section runs when a user hits the return key at the main menu options. This section of the program does the number crunching to generate the output files. The output files are all the result of performing the analysis of coefficient quantization. The roundoff power measurements are run as a separate option under the main menu.

The input to the digital filter analysis tool is a set of numbers that represents the filter coefficients. These coefficients can be input using up to eight significant digits. The output from the digital filter analysis tool are files. These output files contain the results of a filter simulation. Three basic types of files are produced. The magnitude file shows the magnitude response of the filter to different frequencies. The phase file shows the phase relationship for different frequencies. The error file shows the user difference between the best precision available from the input coefficients to the precision available from the number of bits used in the simulation. The error file is a linear difference between the unquantized (single precision) and the quantized version of the magnitude responses. However, some of the error plots in Section 5.3 are calculated as the linear difference from the "design criteria."

The output files that have the magnitude and phase response are listed as two types. The two types correspond to the unquantized (single precision) and the quantized (less than 23-bit fractional number) versions. When the digital filter analysis tool is used, the opening menu displays all the options for the user to select from. Two options allow the user to associate a two digit number with the output files. One of these two options puts a two digit number with the unquantized simulation results. This two digit number is attached to the magnitude and phase plot. The second option associates a two digit number tag onto the quantized output files- magnitude, phase, and the error. By looking at the names of the output files, a user can immediately tell if the file is the unquantized or the quantized simulation results.

4.2.1 Calculation of Unquantized Magnitude and Phase The main routine uses the starting radian frequency point and then hands off control to either the direct magnitude calculation subroutines or the cascade magnitude calculation subroutines. The results from running either one is the value of the transfer function at that point in frequency. The next set of calculations performed are the magnitude and the phase. They are found by classical means as

$$\begin{aligned} \text{Magnitude of H(z)} &= |H(z)| = \left| \frac{\text{numerator of H(z)}}{\text{denominator of H(z)}} \right| \\ \text{Phase of H(z)} &= \theta(z) = \tan^{-1}\left(\frac{\text{imaginary part of H(z)}}{\text{real part of H(z)}} \right) \end{aligned} \tag{4.1}$$

The group delay is also found but not used. This section continues to step through a specified number of spectral points. The arrays built are the single precision (unquantized) magnitude data array and the unquantized phase data array. These two arrays correspond to the unquantized calculations.

With the single precision (unquantized) data arrays built, the next two sections simply write out to the disk the two arrays. However, the magnitude is converted into logarithmic scale before being written to the disk.

4.2.2 Calculation of Quantized Magnitude and Phase Once the unquantized (single precision) data files are written to disk, the quantization analysis section begins. This section will account for the restricted word register lengths to represent the coefficients. The first step is to find the value of all the coefficients in a quantized form.

A subroutine is called (quant) to find the values of the quantized coefficients. This subroutine takes a value specified by single precision format and returns a new value that is dependent on the number of available bits as selected by the user. The new coefficients are used to find the magnitude and phase responses just as in the single precision (unquantized) section. The transfer function is recomputed. Then the magnitude and phase are calculated from the transfer function at the spectral frequency. The magnitude is written in terms of a logarithmic scale. At this point one more calculation is performed. An error measure can be found from the difference between the 24-bit precision magnitude and the further quantized magnitude plots. The error measurement is a linear measure of the error between both magnitude data arrays at the same spectral frequency.

The data arrays for the quantization effects are saved as files. Error checking for out-of-bound numbers is done. Also checks for division by zero is performed and prevented with a message to the user. A useful check for the user is one when the coefficients are quantized. If a quantized version of a coefficient is going to take a value of zero, then a message is sent to the screen telling the user of the program that one of the coefficients was just quantized to zero. More of these messages will occur as less bits are used to represent the coefficients and further degradation in magnitude response can be expected.

4.3 Subroutines Called by the Main Option Menu and Main Computation Section

The Digital Filter Analyzer has many subroutines. This section is to provide an overview of each of the subroutines used that execute tasks. Explanations are given to provide some detail beyond the user's manual level. This section helps to explain how the design was broken down into smaller, manageable sections of code.

4.3.1 Function Quant; Performs the Quantization and Roundoff Computations The main routine calls this function to quantize the coefficients. There are error checks for zero conditions after quantization, and violation of out of bound conditions. Numbers are limited to be within the range ± 1.0 in two's complement format. The routine will provide to the user error messages that display the quantized value and the maximum that is allowed. If the range of the quantized value is beyond the range then the quantized value will be truncated to the minimum or the maximum value.

This routine will return the value of a number in a quantized representation. The effect of quantization and roundoff is similar when applied to number representations. Each source of error degrades the precision to representing the result.

The smallest quantization interval (2^{-B}) is added up until the number is reached. The amount of the smallest incremental steps is added up, then rounded up or down depending on the sign of the number being represented. The integer part is taken from this and then divided by the total number of incremental steps possible with the bits available. The function quant uses the IFIX function in FORTRAN to take the integer part of a number.

$$\text{quantized value} = \frac{IFIX(2^{(\text{number of bits})} * (X) + 0.5)}{2^{(\text{number of bits})}} \text{ for X} \geq 0.0.$$

and

$$\text{quantized value} = \frac{IFIX(2^{(\text{number of bits})} * (X) - 0.5)}{2^{(\text{number of bits})}} \text{ for X} < 0.0.$$

where X is the number being represented.

4.3.2 Roundoff Power Computations The filter is pulsed by a unit impulse and 100 impulse response values are used to compute the roundoff power. The output of the filter is used to find the roundoff power. Two types of output are found. One output accounts for the effects of roundoff and the other output does not account for the effects of roundoff. The error between the outputs is used to compute the roundoff power. The type of structure changes how this approach is applied.

4.3.2.1 Function Roundq; Quantization Routine for the Roundoff Power Calculations. This function does the exact same function as the Quant routine but without the error checking that is done in function Quant. The purpose of this function is to represent a given number to the precision given by the number of bits available for quantization. Out of bound conditions are not checked for in this function (quantized values can be larger than |1.0|. The roundoff error measurements call this function not the quantizer routine.

4.3.2.2 Subroutine Roundoff; Controls the Roundoff Computations. The inputs to this subroutine are the coefficients, type of filter and number of bits available. The subroutine will direct the flow of the program into one of two types of digital filters structures. The two structures are the direct structure and the cascade structure.

Once the control is passed back to the subroutine, another option for the user to enter is provided. When a cascade structure is being used, then the user will have the option to

change the order of the cascade sections. This provision is applicable in the reduction of roundoff power. Control is returned to the main menu.

4.3.2.3 Subroutine Roundcascade; Computes Roundoff Power Generated By Cascade Sections. This subroutine finds the output for the cascade structure in both the single precision (unquantized) and the quantized ($(B+1)$ bit fixed point) structures. The difference between the two outputs is used to estimate the power due to roundoff noise.

The routine requires that the cascade design is used. Therefore, an iterative structure is used to compute the responses. The impulse response is found from the transfer function using the first 100 output points. The impulse response is first computed without the effects of the roundoff noise. Then, the routine computes the impulse response for the first 100 points that account for the effects of roundoff noise. The number of bits available are input by the main menu. The use of 100 points will generally provide time for the impulse response to vanish. The algorithm that is used is difficult to understand unless the form of the cascade section is known. Each node has a new value with each iteration in the impulse response.

Figure 4.1 shows the structure to implement the algorithm. The value of P[n] is computed first, then the others follow as in the following set of equations. The parameter K steps though cascaded second-order sections to the input that was generated by the prior section's output.

$$P[N] = X[N] + (P1[N] * a_2) + (P2[N] * a - 3) \tag{4.2}$$

$$Y[N,K] = (P[N] * b - 1) + (P1[N] * b_2) + (P2[N] * b_3) \tag{4.3}$$

The summation nodes of $P1[n]$ and $P2[n]$ are updated with the value one discrete time previous to the present discrete time. This provides the node with the values to be used during the next iteration. This process gives a result in a two dimensional array $Y[N,K]$. The second dimension is a means to keep the output of each cascade section. The output of each cascade section becomes the input for the next cascade section.

The same algorithm is used again except the results from each multiply are sent to the roundoff routine (accounts for roundoff errors). The roundoff routine simulates the effect of having only a limited number of bits available to represent the result from the multiplies.

Once the impulse response is found for the first 100 points in the cascade filter accounting for roundoff, roundoff power can be estimated. To find the roundoff power, the mean for

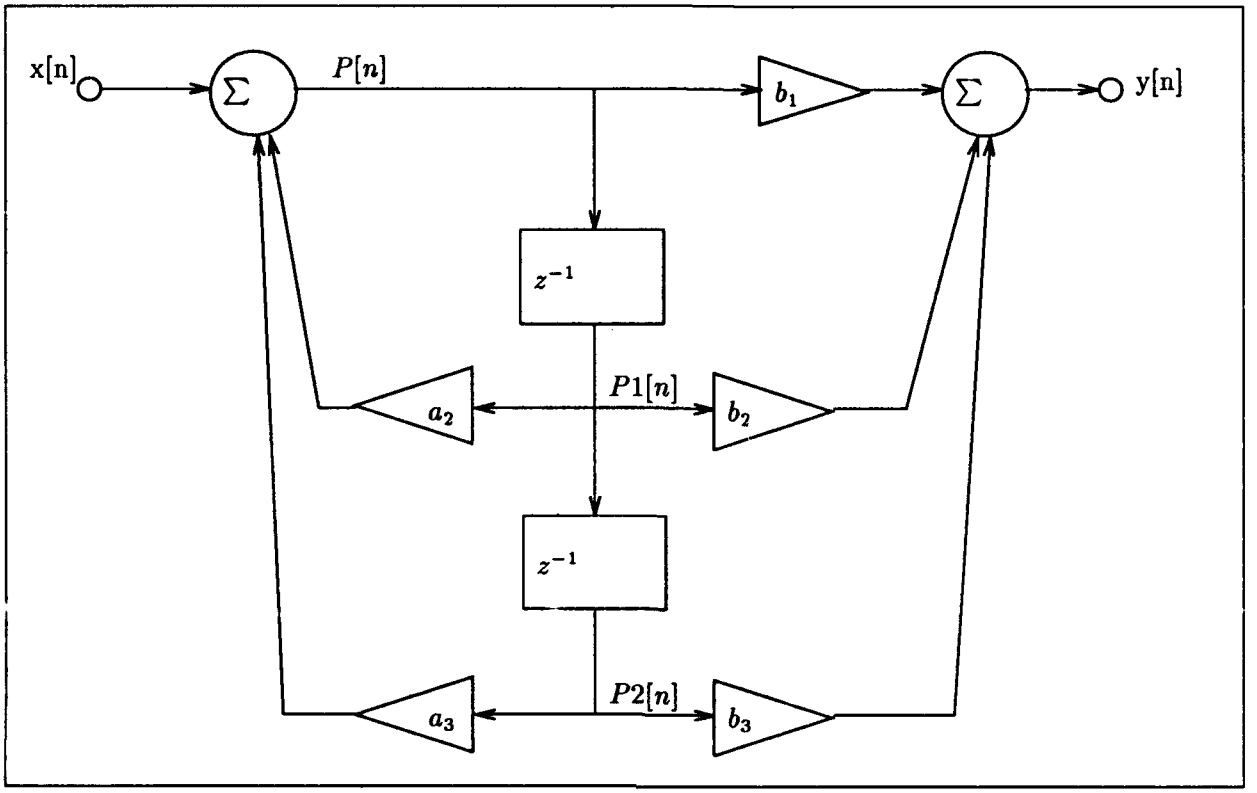


Figure 4.1. Cascade structure with Direct Form II realization for the second-order section.

the error between the two impulse responses is found. The impulse response from the cascade filter without accounting for the effects of roundoff is subtracted from the the impulse response from the cascade filter that does account for the effects of the roundoff errors from multiplication. The expected value of the difference is found. A recursive estimator is used since it can apply to many applications [20], as shown in Equation 4.4. A pictorial view of the implementation is given in Figure 4.2. The equation to use the estimator that finds the current mean $y[n]$ is as follows:

$$\bar{y}[n] = \bar{y}[n-1] + \frac{1}{N}(x[n] - y[n-1]) \tag{4.4}$$

The noise power is found by the use of [14:298]

$$\text{error} = (h_k - \hat{h}_k) \tag{4.5}$$

h_k = impulse resonse calculated without rounding.

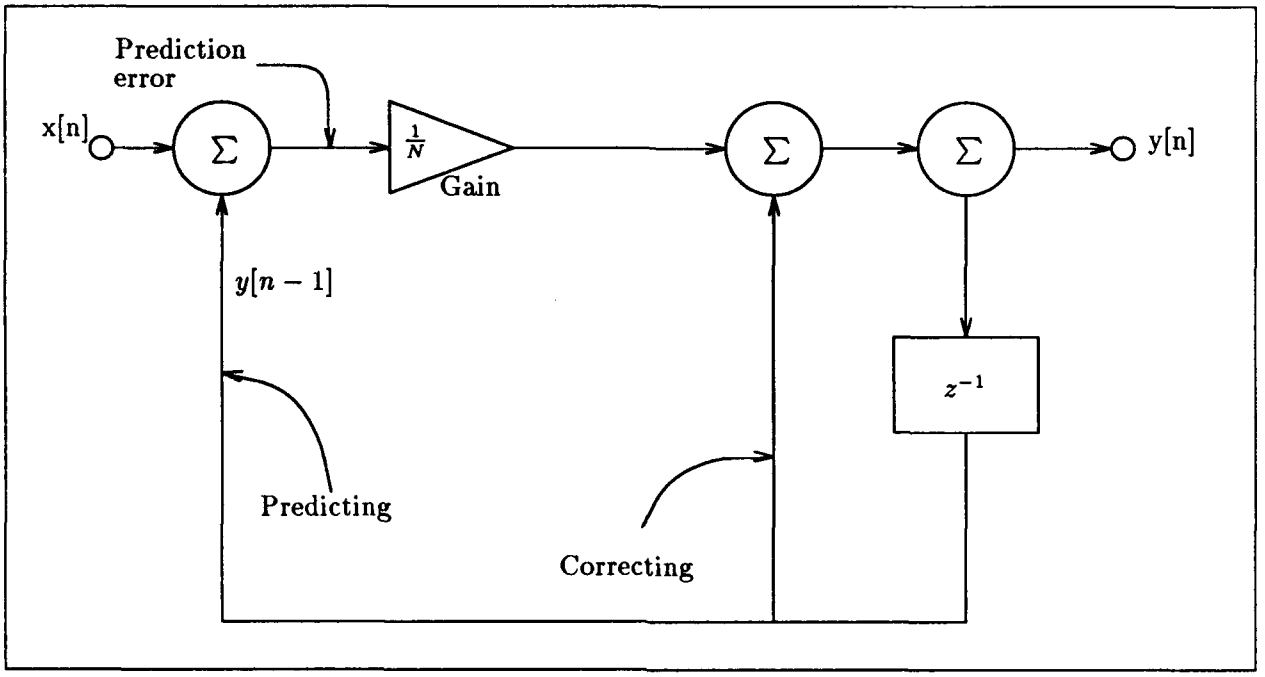


Figure 4.2. Recursive Mean Estimator Implementation

$\hat{h}_k$ = impulse response calculated with rounding.

and

$$\text{roundoff power} = \frac{1.0}{(N-1)} \sum_{n=1}^{N} (\text{error} - \mu)^2. \tag{4.6}$$

where μ is the mean value of the 100 points in the error sequence.

The roundoff power is sent to the screen. The order of the cascade sections can be changed by the user. After a change in the order of the cascade sections, the roundoff power is computed and displayed again to show the effects of changing the order of the second-order cascade sections.

4.3.2.4 Subroutine RoundDirect; Computes the Roundoff Power in Direct Form I. This subroutine is called by the subroutine roundoff. The purpose of this routine is to find the impulse response of the filter for the first 100 sample points. Then the routine finds the impulse for a Direct Form I digital filter that accounts for the effects of roundoff from the multiplications. The same approach is used for the Direct Form I to find the power in the roundoff noise as in the approach to find the power in cascade forms. Equations 4.4 and 4.6

are used to find the roundoff noise power generated by the filter. This information is used to find the power in the roundoff error as it applied to the current filter structure.

The use of 100 sample points only applies to the infinite impulse response forms. The use of Direct Form I for the FIR case need only contain twice as many sample points as there are delay elements. Figure 4.3 shows the form of an infinite impulse response filter with second-order feed forward and second-order feed back loops.

The value of the roundoff power is presented to the user on the screen. Once the analysis for the roundoff power is done for the Direct Form I, control is returned to the main menu.

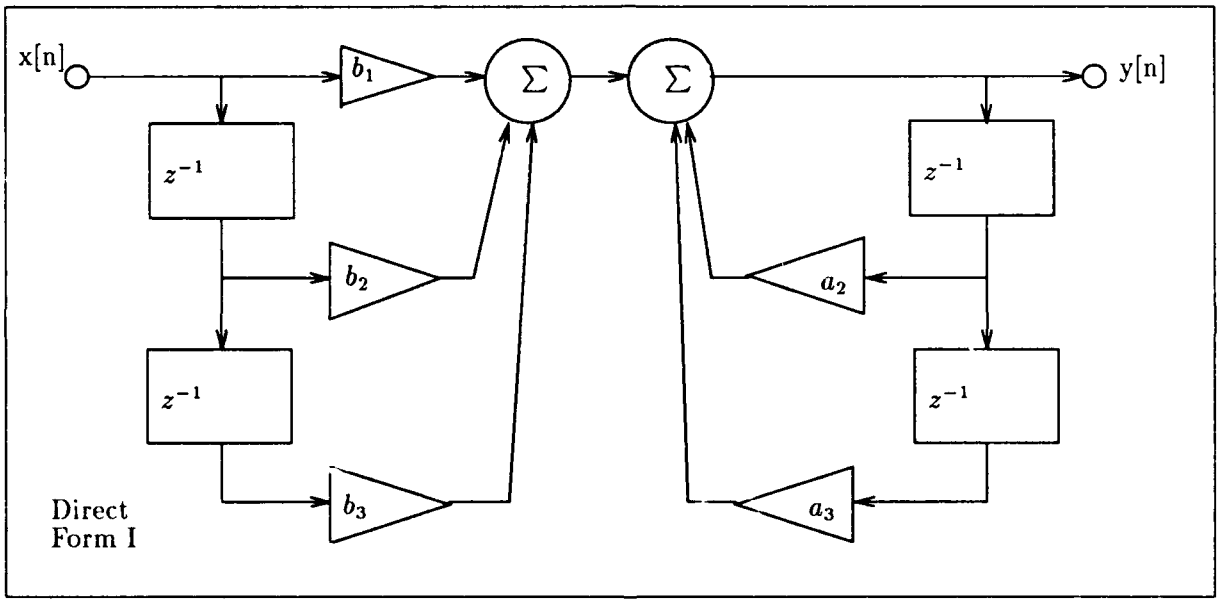


Figure 4.3. Infinite Impulse Response structure, second-order feedback and second order feedfoward sections.

4.3.3 Subroutine ChangeCascade Order; Allows Manipulation of the Cascade Sections for the Numerator and the Denominator. This subroutine is called by the subroutine round cascade. Once the calculation for the roundoff power is completed, the user is presented with the option to change the order of the cascade sections. By manipulating the order of the cascade sections, the user will be able to maintain the same transfer function of the digital filter, but the user will be able to manipulate the resulting roundoff power developed within the filter.

The basic idea is to manipulate the order of the second-order sections. Since it has been shown that the order of the transfer functions will not change the overall system response (assumes unquantized coefficients), the designer is assured of frequency performance goals. Therefore, the designer is further able to refine the design by finding the lowest possible roundoff power measurement. The process is best done by matching the pole zero pairs that are close to each other. Then the order for the filter sections should start with those pole zero pairs that are closest to the unit circle [5]. As shown in Figure 4.4, the pairing procedure can be done by a graphical means. This can serve a starting point. The designer can then try to change the order of the sections to minimize the roundoff power.

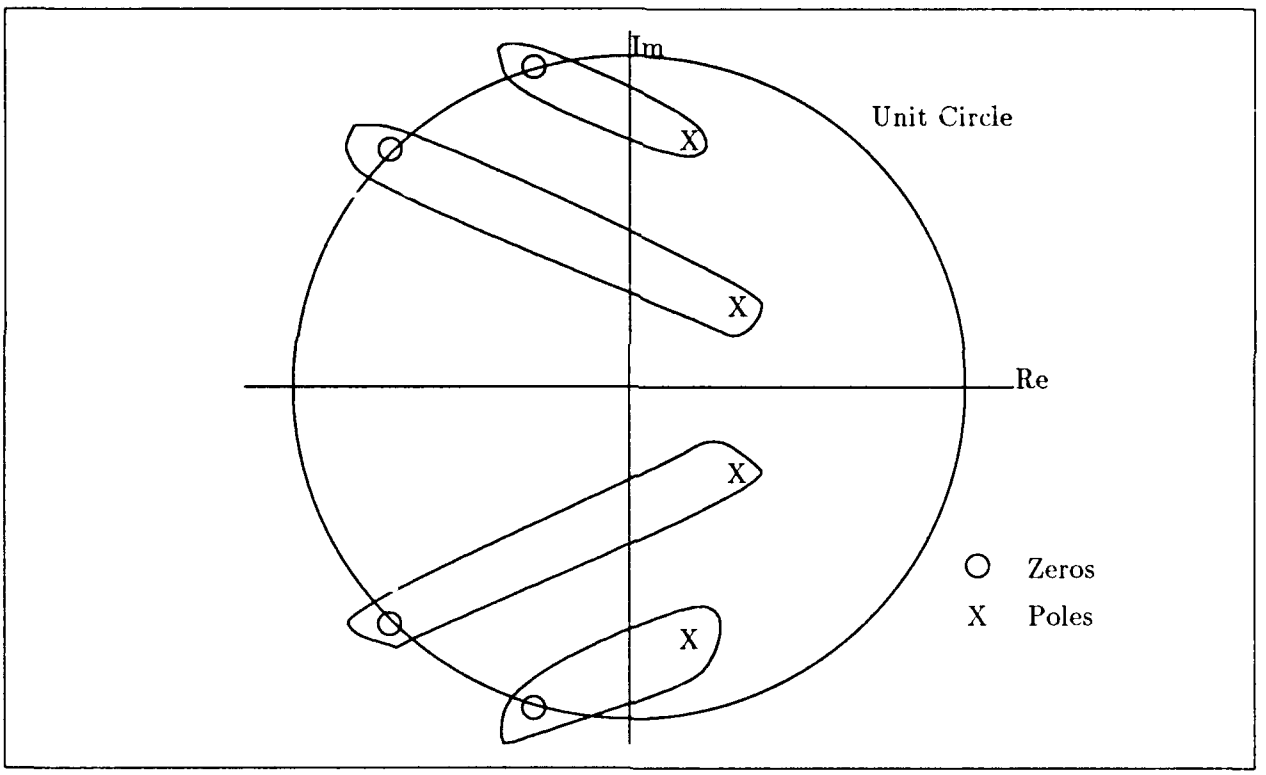


Figure 4.4. Pole-zero plot for a fourth-order filter showing pairing of most reasonable pairs.

When this subroutine is entered, the coefficients for the cascade designed filter are sent to the screen. Once the coefficients are sent to the screen, the roots of the cascade sections are found and sent to the screen. The user can view the roots of each section in both the numerator and the denominator. By seeing the location of these roots, the user can choose the best order to place the cascade sections.

The roots are normally complex conjugates of the second-order polynomial. With this in mind the equation

$$\text{roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

is used to find the roots. If the user wants to separate the complex conjugates by splitting up the conjugate pair, the re-building of the second-order polynomial will have the following form:

$$(z + (\alpha + j\beta))(z + (\gamma + j\delta)) \tag{4.7}$$

where the real part and imaginary part of the roots are no longer complex conjugates. By expanding Equation 4.7 we obtain,

$$(z^2 + (\alpha + j\beta)z + (\gamma + j\delta)z + j\beta\gamma + j\delta\alpha - \beta\delta). \tag{4.8}$$

This can also be written by combining terms as,

$$(z^2 + (\alpha + \gamma + (\beta + \delta)j)z + (\beta\gamma + \delta\alpha)j - \beta\delta). \tag{4.9}$$

Two coefficients in Eq. 4.9 have real and imaginary parts. Separating the complex conjugates will lead to a more complex heuristic design process for the designer.

Since the complex conjugate no longer will zero the imaginary terms shown in Eq. 4.9, the resulting second-order section will have complex coefficients. This form can not be implemented by any realization directly. The only solution is to use higher order sections to incorporate the complex conjugates. The final lowest order polynomial may indeed be a direct form realization. In this case, the designer will more than likely achieve better results with lower cost in implementation by adding a bit to the word length and working with the best performance achievable by the effects of the roundoff design. This clearly shows how a designer is faced with the trade-offs of complexity and higher performance.

Fortunately, the pole-zero plot will have the placement of the complex conjugates in a form where the designer can always choose the pair closest to the real axis and the unit circle. The designer must keep the complex conjugates as a pair in the numerator and the denominator sections. The digital analyzer tool does this. The designer can move the complex conjugate pairs as a unit in the cascade designed filter. By so doing, the rebuilding of the second-order sections is guaranteed to have real coefficients, a requirement for design.

4.3.4 Summary The method to quantize numbers for representing them in binary format with the decimal point to the left of the most significant bit was given. The quantization process is used to simulate the coefficient quantization and aid in the simulation of roundoff noise. The process to calculate the roundoff power was given. The subroutines to perform the calculations were reviewed. Finally, the method to rearrange the second-order sections to reduce the roundoff power generated within the structure was reviewed. Appendix C contains further detail information on subroutines not mentioned in this chapter.

V. Experimental Results From The Digital Filter Simulator Tool

5.1 Overview of Chapter

In this chapter, the results from using the digital filter simulator tool are presented. A comparison of roundoff power is done using theoretical means and the digital filter analyzer tool. Results from the comparison use various lengths for the representation of coefficients to establish a pattern in the roundoff power measurement.

Analyses of FIR and IIR filters in Direct and Cascade forms are presented. Filter responses are verified against other works. The mechanism used to verify the results is to visually compare between the magnitude response plots. This section is not concerned how the coefficients are generated rather, on the effects of specific realizations.

The plots in this chapter will show the number of bits used for the precision. The number of bits used for the representation is $(B + 1)$ where B is the number of bits for precision and the 1 is for the sign of the number being represented. The coefficients for the filter are assumed to be in two's complement representation.

5.2 Comparison Of Roundoff Power Computations.

The results of the theoretical calculations are compared to the results of the simulator. Conclusions are then drawn from the comparison.

The results from the digital filter simulator tool will provide sound computation of the roundoff power generated within the filter. A simple cascade section is implemented by the simulator tool. The software simulator estimates the power generated by the roundoff errors. The theoretical computation of the same cascade section is done in Section 3.5.3. The results are compared in Table 5.2.

For purposes of this comparison, the second-order cascade realization is used. The cascade form can be from the transfer function

$$H(z) = \frac{0.75}{(z - 0.4)(z - 0.5)} \tag{5.1}$$

as

$$H(z) = \left(\frac{0.0 + 0.0z^{-1} + 0.75z^{-2}}{1 - 0.9z^{-1} - (-0.45)z^{-2}} \right) \quad . \tag{5.2}$$

The digital filter simulator tool used the coefficient file shown in Table 5.1 to generate the output roundoff power measurements for different wordlengths. The wordlengths are specified by the number of bits used in the simulation. Table 5.2 compares the results from the theoretical calculations and the simulated calculations.

Table 5.1. Coefficients for the Cascade section used for roundoff power calculation comparisons.

Coefficient Position	Coefficient Value
Section Number One	
a_1	1.00
a_2	0.90
a_3	-0.45
b_1	0.00
b_2	0.00
b_3	0.75

The Difference column is found by taking the difference between the theoretical and simulation results.

Simulating the effects of the roundoff power is a quicker method than using the contour integral. The results, when compared to the theory, are close (in the same order of magnitude). Table 5.2 shows the roundoff from theory is larger and smaller than the roundoff generated by the simulator. The probability distribution for the error is assumed to be a linear distribution that causes the theory to find a consistent roundoff power figure. The simulator will find and account for the actual error produced within the simulation roundoff process.

Table 5.2. Comparison of roundoff power measurements for different wordlengths computed by theoretical means and by the software simulator.

Number of Bits	Theory	Simulation	% Error	Less Power in Sim.?
22-bits	0.507E-13	0.2317E-13	78%	no
20-bits	0.8124E-13	0.8956E-13	9%	no
18-bits	0.1300E-11	0.5802E-11	77%	no
16-bits	0.2080E-10	0.1180E-10	43%	yes
14-bits	0.3328E-09	0.3486E-09	5%	no
12-bits	0.5324E-08	0.5741E-08	7%	no
8-bits	0.1326E-05	0.1037E-05	24%	yes
6-bits	0.2181E-04	0.1867E-04	14%	yes

Table 5.3. Unquantized and quantized coefficients for an FIR filter.

Impulse Value	Coefficient	Quantized Coefficient to 8-bits
h[0]=h[28]	1.359657E-3	0.00000000
h[1]=h[27]	-1.616993E-3	0.00000000
h[2]=h[26]	-7.738032E-3	-7.8125E-3
h[3]=h[25]	-2.686841E-3	0.00000000
h[4]=h[24]	1.255246E-2	1.56250E-2
h[5]=h[23]	6.591530E-3	7.81250E-3
h[6]=h[22]	-2.217952E-2	-2.34375E-2
h[7]=h[21]	-1.524663E-2	-1.56250E-2
h[8]=h[20]	3.720668E-2	3.90625E-2
h[9]=h[19]	3.233332E-2	3.12500E-2
h[10]=h[18]	-6.537057E-2	-6.2500E-2
h[11]=h[17]	-7.528754E-2	-7.8125E-2
h[12]=h[16]	1.560970E-1	1.56250E-1
h[13]=h[15]	4.394094E-1	7.81250E-3

The simulator computes the roundoff power by using 100 samples of the impulse response from the transfer function $h[n]$. Generally, for the IIR type filters, 100 iterations will allow most of the impulse response to be close to zero. A more complicated input/output signal and extending the iterations to reach a minimum value may have given a tigher agreement. Details for the computation can be found in Section4.3.2.3.

5.3 FIR Example With Linear Phase Showing Results with Coefficients Quantized.

A simple illustration is presented of an FIR filter that was designed with the Parks-McClellan design technique. The design was to meet the following constraints:

$$0.99 \leq \mid H(e^{j\omega}) \mid \leq 1.01, \quad 0 \leq \omega \leq 0.4\pi$$

$$\mid H(e^{j\omega}) \mid \leq 0.001(-60dB), \quad 0.6\pi \leq \omega \leq \pi$$

Table 5.3 shows the resultant coefficients and the corresponding 8-bit quantized version. Since this is an FIR filter with even symmetry about the center of the impulse response, the phase is linear. The results show that even symmetry is maintained after quantizing the coefficients.

Figure 5.1 through Figure 5.19 show the results of the coefficient quantization for eight cases: 24-bits (unquantized), 16-bits, 14- bits, 13-bits,12-bits, 10-bits, and 8-bits, and 6-bits. The design criteria is met in the 16-bit, 14-bit, and 13-bit representations. However, the 12-bit magnitude plot shows the stop band criteria of 60dB attenuation is not met. Figure 5.5 shows that some frequencies are only 53dB down. Figure 5.7 (8-bits) of the magnitude shows that the stop band is more than 10 times the desired level of output.

Figure 5.8 (6-bits representing the coefficients) shows degradation in the transition region to the stop band region verses the 24-bit plot. This plot does have about 13% of the frequency band in the transition region. This example shows how the transition region holds it form during the quantization process. Once beyond 60% of the frequency band, the stop band does hold 20dB down.

The digital filter simulator tool did truncate 14 of the 28 filter coefficients to a value of zero when only 6-bits were available to represent the filter coefficients. Since the zeros that were truncated were very close to the value of zero, the actual amount of change is still limited to the accuracy of the number of bits available. In reality, all the coefficients were truncated to accuracies limited to the number of bits. The coefficients close to zero may have the distance moved to the new coefficient location (zero for this case) that is actually less distance than other coefficients that did not suffer truncation to zero.

The error and phase plots using 6-bits to represent the coefficients are omitted, since they show similar information. The error plots in Figure 5.9 through Figure 5.14 show the error corresponding to either the design criteria or the 24-bit version of the magnitude response. In particular, Figures 5.10 and 5.12 show the linear error from the design criteria. The first section of Figures 5.10 and 5.12 correspond to the pass band region; the error past 1.5 radians corresponds to the error in the stop band region. The transision region is set to zero for these two plots.

The error shows how much deviation occurs from the unquantized plots (shows the deviation from optimal conditions). Another type of error plot shows the deviation from the design criteria (shows the absolute error). The deviation in the stop band region consistently worsens as the number of bits is decreased.

Also presented are the phase plots (Figure 5.15 through Figure 5.19) for all cases except the 6-bit version. This shows that the filter will maintain linear phase. The is due to the symmetry of the locations of all the zeros in the z-plane. This is one area that maintains its benefits even under the effects of coefficient quantization.

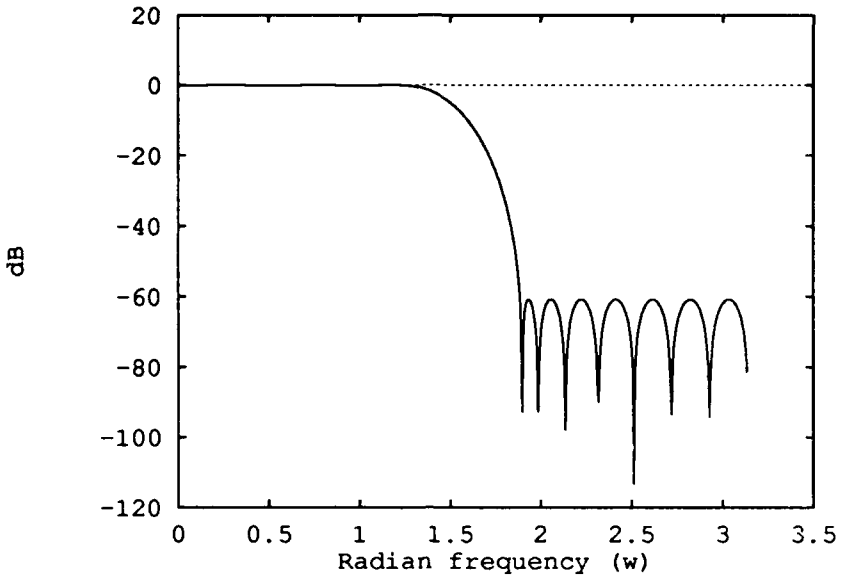


Figure 5.1. Unquantized Magnitude Plot, Full Single Precision, representation of coefficients in Table 5.3.

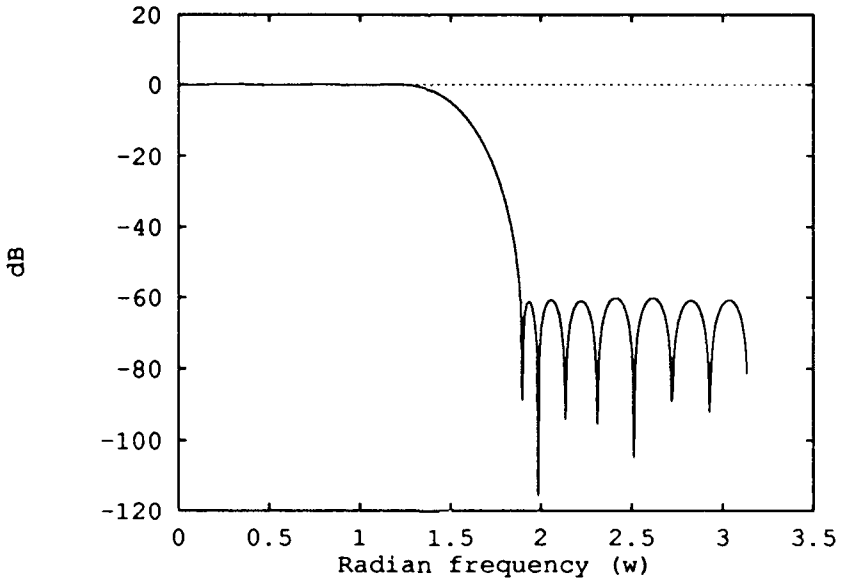


Figure 5.2. Quantized Magnitude Plot, 16-Bits Precision, representation of coefficients in Table 5.3.

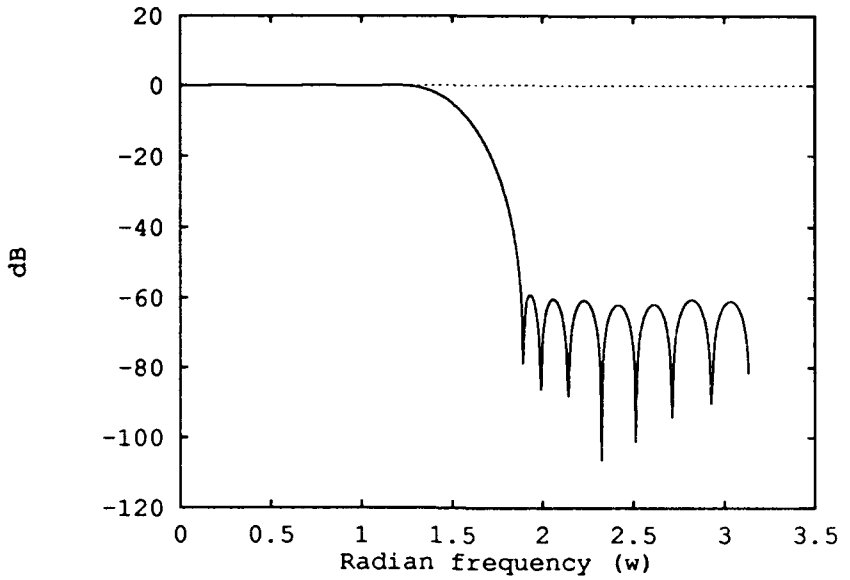


Figure 5.3. Quantized Magnitude Plot, 14-Bits Precision, representation of coefficients in Table 5.3.

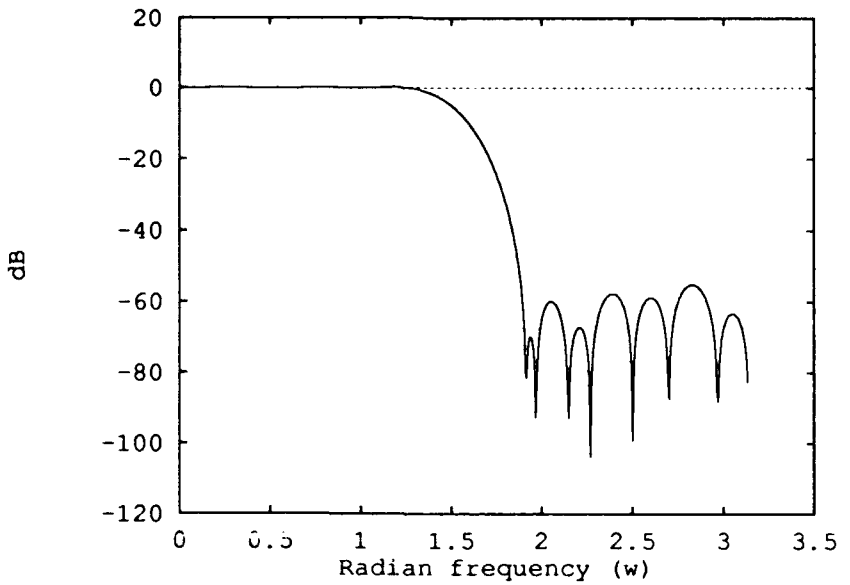


Figure 5.4. Quantized Magnitude Plot, 13-Bits Precision, representation of coefficients in Table 5.3.

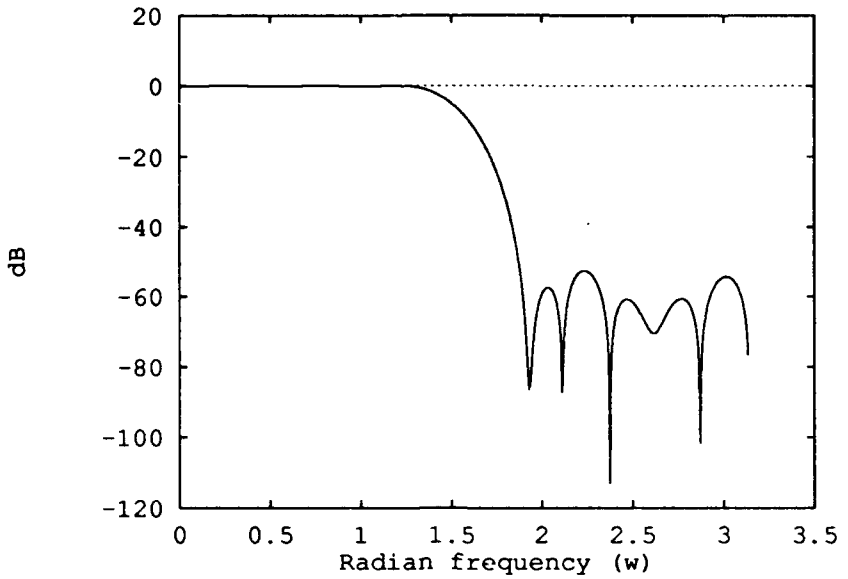


Figure 5.5. Quantized Magnitude Plot, 12-Bits Precision, representation of coefficients in Table 5.3.

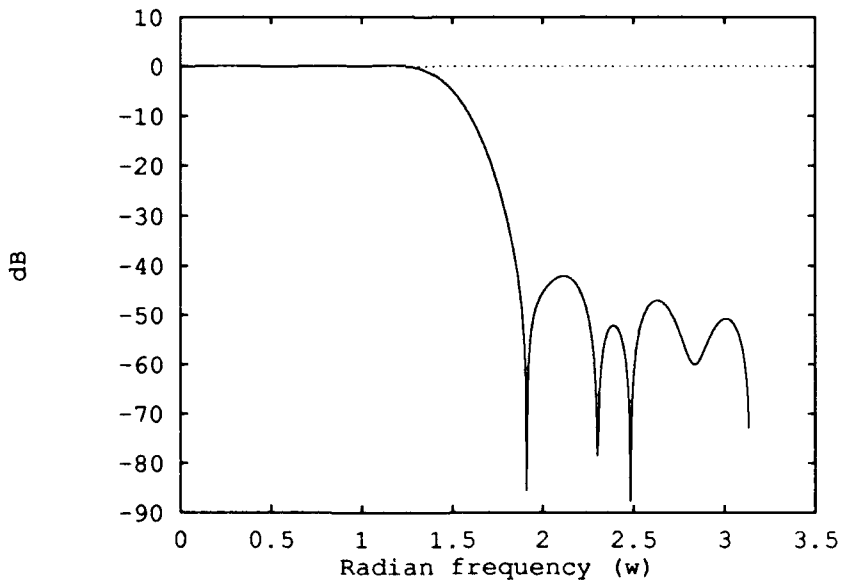


Figure 5.6. Quantized Magnitude Plot, 10-Bits Precision, representation of coefficients in Table 5.3.

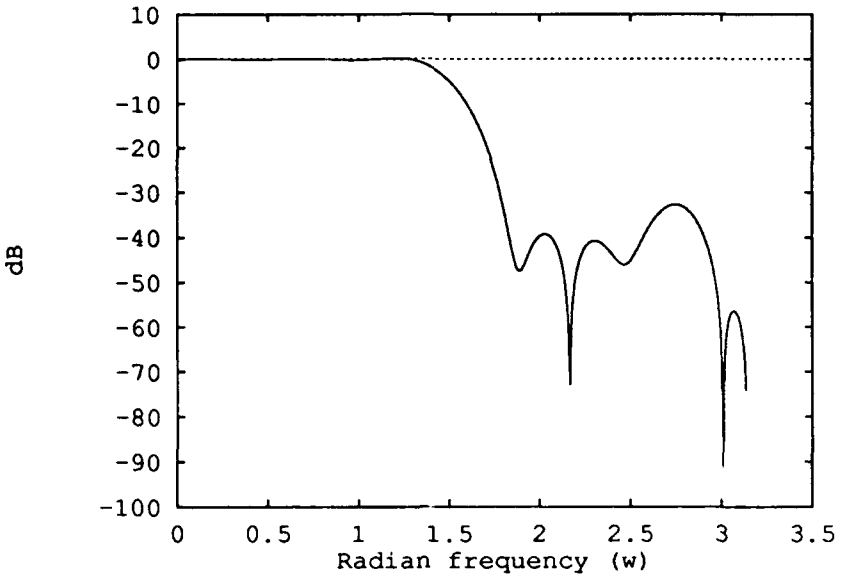


Figure 5.7. Quantized Magnitude Plot, 8-Bits Precision, representation of coefficients in Table 5.3.

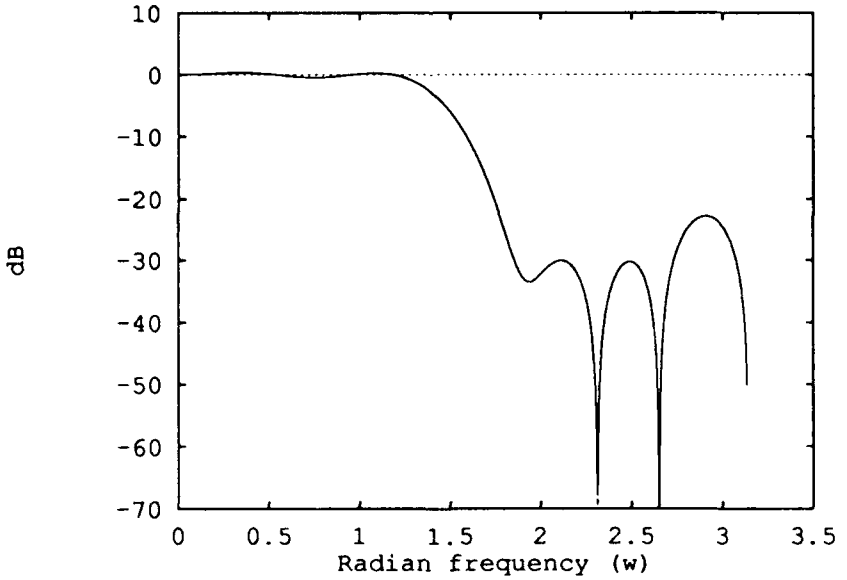


Figure 5.8. Quantized Magnitude Plot, 6-Bits Precision, representation of coefficients in Table 5.3.

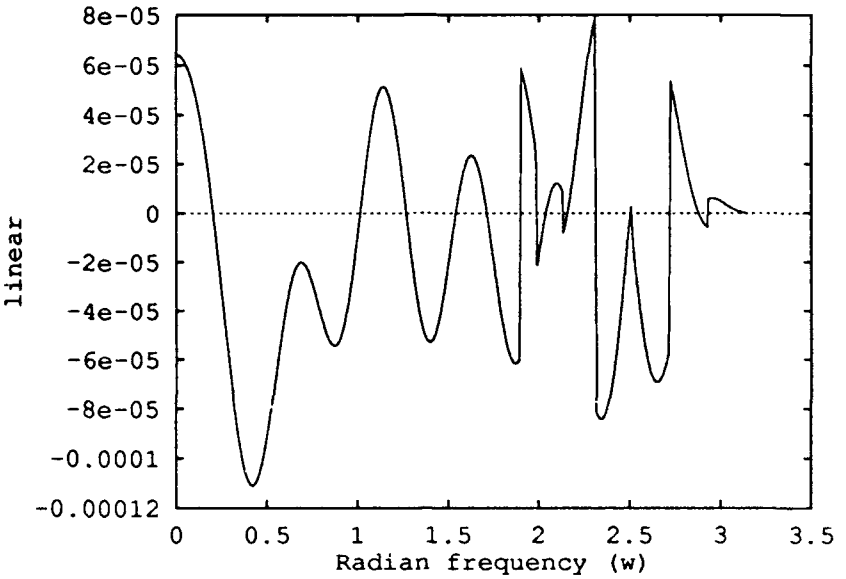


Figure 5.9. Linear error (difference from the single precision version), 16-Bits precision, representation of coefficients in Table 5.3.

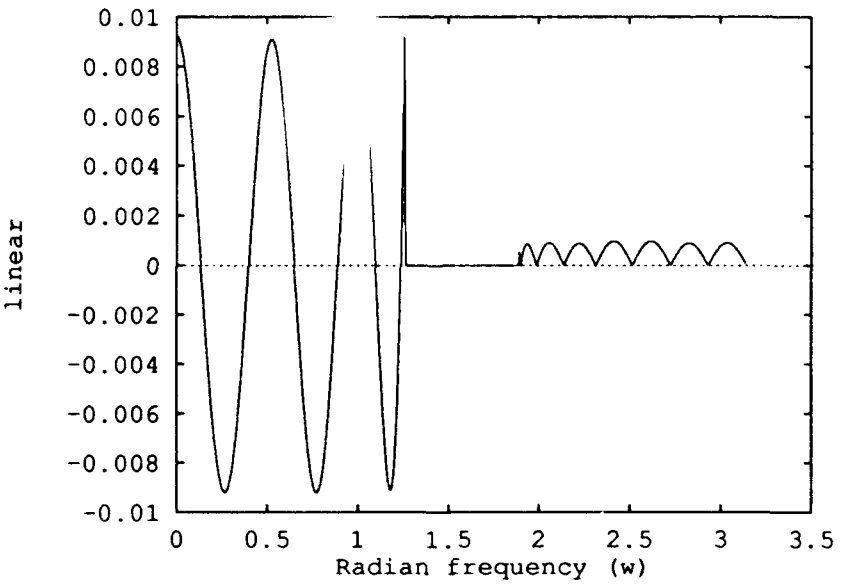


Figure 5.10. Linear error (difference from the design criteria) 16-Bits precision, pass band and stop band shown, representation of coefficie ; in Table 5.3.

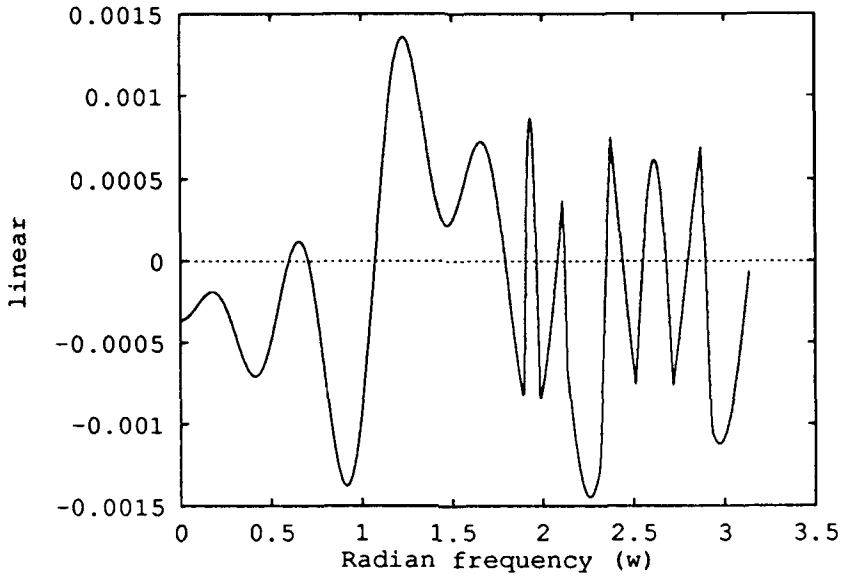


Figure 5.11. Linear error (difference from the single precision version), 12-Bits precision, representation of coefficients in Table 5.3.

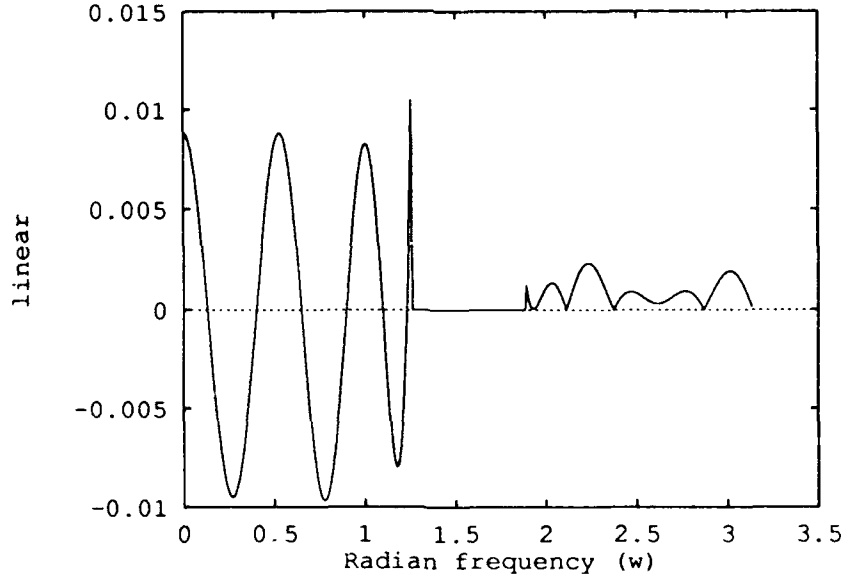


Figure 5.12. Linear error (difference from the design criteria), 12-Bits accuracy, pass band and stop band shown,representation of coefficients in Table 5.3.

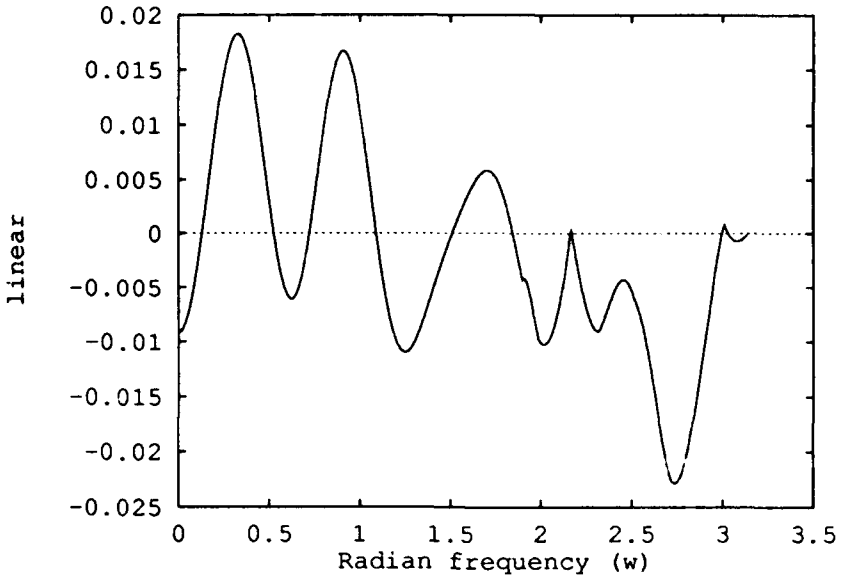


Figure 5.13. Linear error (difference from the single precision version), 8-Bits accuracy, representation of coefficients in Table 5.3.

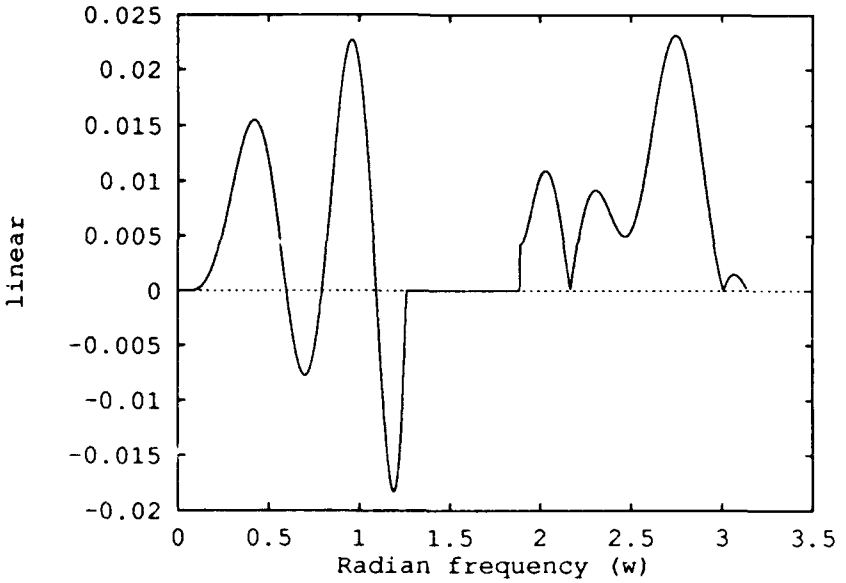


Figure 5.14. Linear error (difference from the design criteria), 8-Bits accuracy, representation of coefficients in Table 5.3.

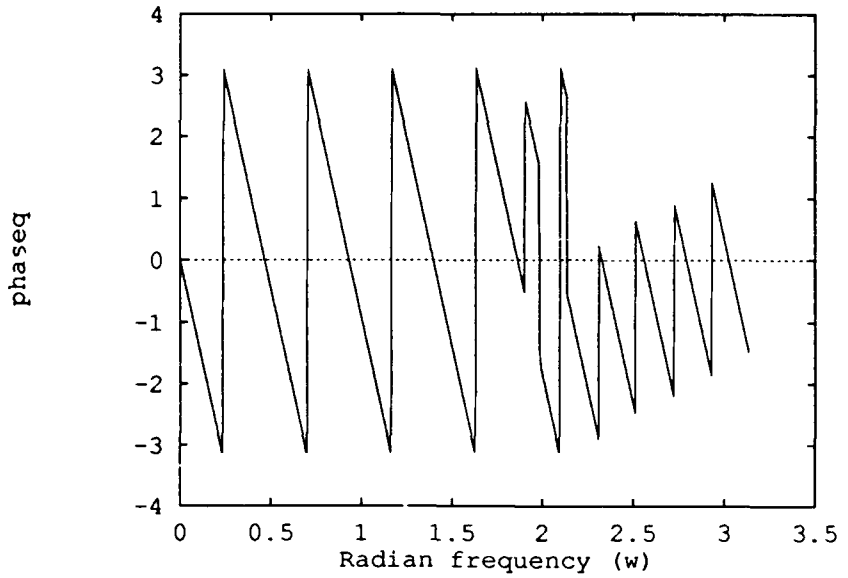


Figure 5.15. Phase Response 16-Bits accuracy, representation of coefficients in Table 5.3.

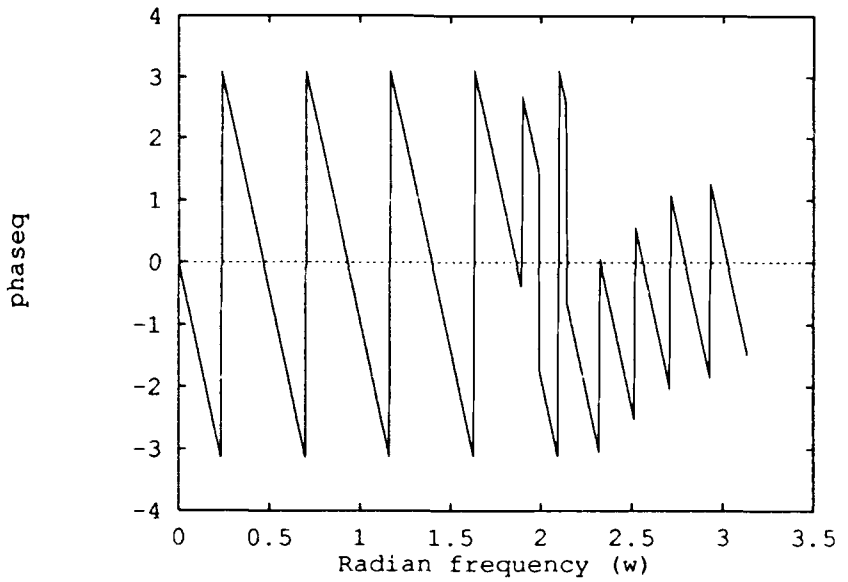


Figure 5.16. Phase Response, 14-Bits precision, representation of coefficients in Table 5.3.

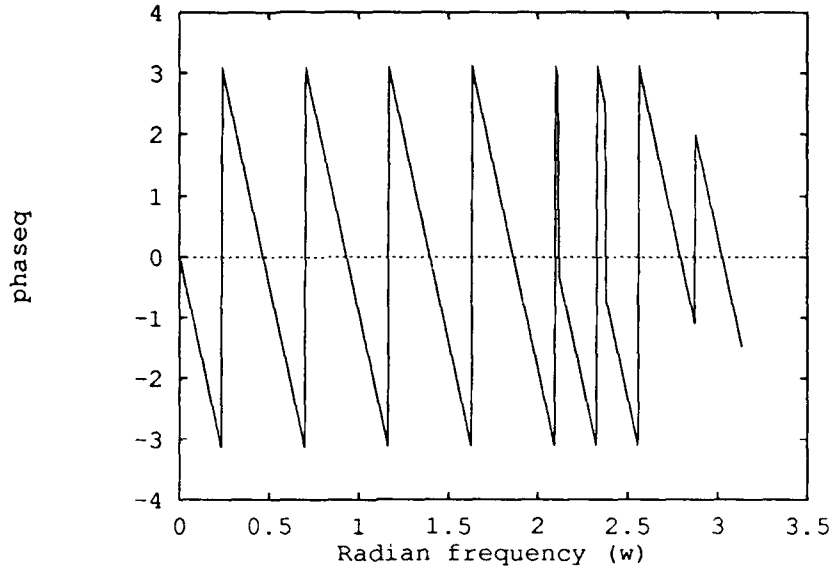


Figure 5.17. Phase Response, 12-Bits precision, representation of coefficients in Table 5.3.

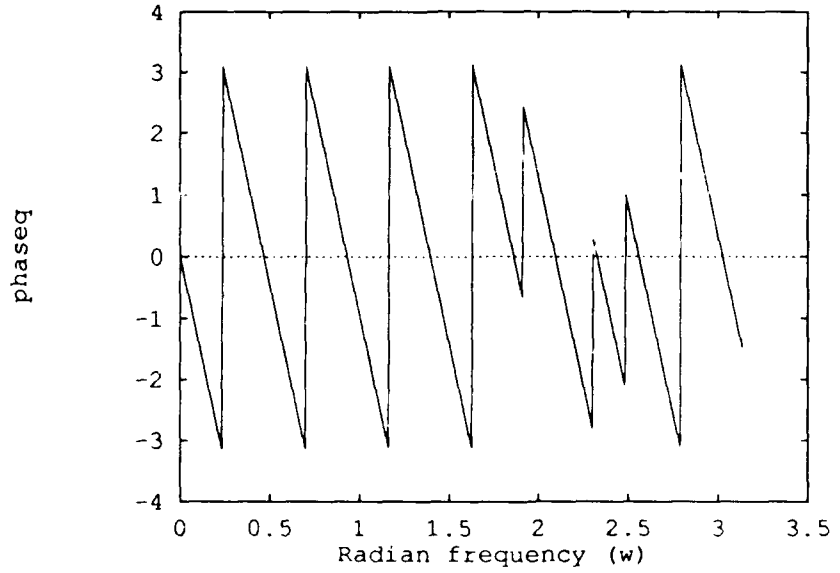


Figure 5.18. Phase Response, 10-Bits precision, representation of coefficients in Table 5.3.

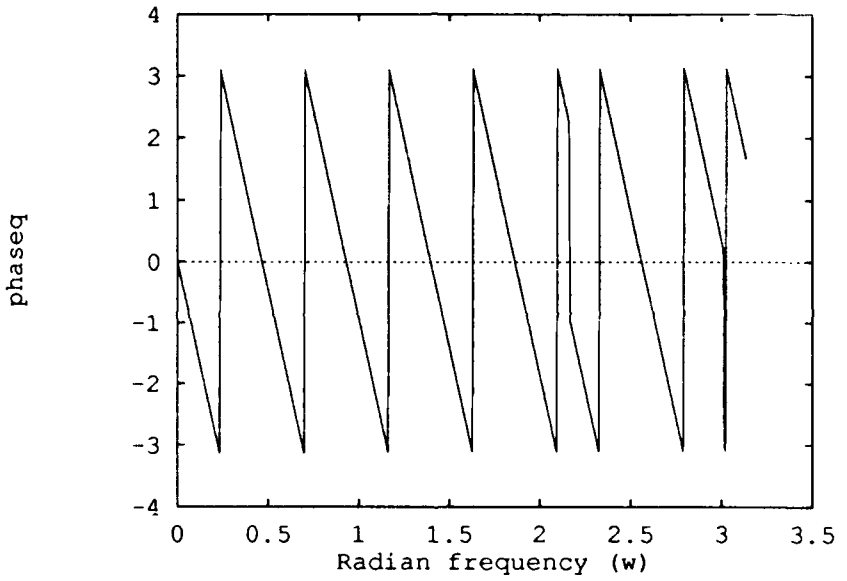


Figure 5.19. Phase Response, 8-Bits precision, representation of coefficients in Table 5.3.

5.4 *FIR Example Implemented in Direct and Cascade Form*

Consider the effects of changing the structure for implementation of a digital filter. Consider the following difference equation.

$$y[n] = 1.0x[n]+2.578793x[n-1]+3.497540x[n-2]+2.507401x[n-3]+1.265625x[n-4]. \quad (5.3)$$

Table 5.4 shows the coefficients and the corresponding position in the filter.

Table 5.4. Coefficients to Begin the Investigation of an FIR Filter.

Coefficient Position	Coefficient
b_1	1.0
b_2	2.578793
b_3	3.497540
b_4	2.507401
b_5	1.265625
a_1	1.0

This example shows that the digital filter simulator tool provides accurate results when each structure is used and the conversion utility is exercised. Direct and cascade structures are used to compute output files. Furthermore, the conversion utility is exercised to see the identical magnitude plots. The files that hold the filter coefficients are the following:

- direct form coefficients not normalized
- direct form coefficients normalized
- cascade form coefficients not normalized
- cascade form coefficients normalized
- conversion from cascade to direct form, normalized.

The coefficients shown in Table 5.4 are provided to digital filter analyzer tool. These coefficients are in the direct form. The coefficients required to implement the filter in cascade form are also computed for the simulator. The simulator computed the rest of the filter files using the normalization routine. The conversion utility found the coefficients in direct form from using the cascade coefficients. Comparisons are made to validate the conversion utility.

The list of output plots uses the filter coefficients in the form stated and the number of bits for coefficient representation. The plots are as follows:

1. 24-bit Magnitude Plot, conversion from cascade structure to direct structure, low pass FIR.
2. Quantized Magnitude Plot, direct structure 16-Bits.
3. Quantized Magnitude Plot, direct structure 8-Bits.
4. Quantized Magnitude Plot, cascade structure 16-Bits.
5. Quantized Magnitude Plot, cascade structure 8-Bits.
6. Magnitude Plot, conversion from cascade structure to direct structure using 16-bits.

The digital filter simulator tool was used to create the normalized files. Table 5.5 shows the coefficients used to start this design example and the results of the normalization process.

Table 5.5. Coefficients in Non-normalized and Normalized States for the FIR Filter In Direct Form I.

Coefficient Position	Coefficient	Normalized Coefficient
b_1	1.0	0.2859152555
b_2	2.578793	0.7373162508
b_3	3.497540	1.0000000000
b_4	2.507401	0.7169041634
b_5	1.265625	0.3618614674
a_1	1.0	0.2859152555

The original design produced coefficients in the direct form. In order to implement the same filter in cascade form, the poles and zeros are needed. The coefficients were found for second-order sections to implement the cascade structure. Since some of the resultant coefficients for the cascade sections were greater than one, the normalization routine was used to create the normalized coefficients shown in Table 5.6. Then the conversion process was used to convert the cascade structure into a direct structure. The result of this conversion process is shown in Table 5.7. The simulator was run using cascaded second-order filter sections. The question to find out is if the coefficients in Table 5.4 will generate the same magnitude plot as the coefficients generated by the conversion process. Note the success of the digital filter simulator tool by viewing Figure 5.20 through Figure 5.28.

Figure 5.20 shows the result of the conversion process. Compare this with Figure 5.21 and with Figure 5.23. Note the magnitude plots appear exactly alike. This shows the digital

filter simulator tool provides consistent results with each of the filter structures and with the conversion process.

By careful examination of Figure 5.22, the use of 8-bits seems to drop the stop band by 2dB when compared to the 24-bit plot. That would mean to a designer that the use of 8-bits does enhance the response of the filter over the use of more accurate coefficient representations. The error plots show the degradation in the filter's performance when fewer bits are chosen.

The transfer function of the direct form coefficients from Table 5.5 implemented in second-order sections is

$$H(z) = \frac{(1.0 + .556231z^{-1} + .81z^{-2})(1.0 + 2.02254z^{-1} + 1.5625z^{-2})}{(1.0 - 0.0z^{-1} - 0.0z^{-2})(1.0 - 0.0z^{-1} - 0.0z^{-2})}. \tag{5.4}$$

This transfer function has the coefficients in a non-normalized form. When the coefficients are normalized, the program produces the filter coefficients as shown in Table 5.6.

Table 5.6. Coefficients for the cascade second-order sections before and after normalization.

Coefficient Position	Coefficient Value	Normalized
Section Number One		
b_1	1.0	0.4944277704
b_2	.556231	0.2750160694
b_3	.81	0.4004864991
a_1	1.0	0.4944277704
a_2	0.0	0.0000000000
a_3	0.0	0.0000000000
Section Number Two		
b_1	1.0	0.4944277704
b_2	2.02254	1.0000000000
b_3	1.5625	0.7725433707
a_1	1.0	0.4944277704
a_2	0.0	0.0000000000
a_3	0.0	0.0000000000

The digital filter analyzer is then used to convert the set of cascade second order coefficients into a direct form. Table 5.7 shows the results of that conversion process.

It is shown from Table 5.7 that the results differ by a gain factor. The magnitude plots also show the similarity in each set of coefficients used for a filter. This validates the

Table 5.7. Coefficients in direct form before and after the conversion process for an FIR Filter.

Coefficient Position	Starting Coefficient Values	Conversion Results
b_1	1.0	0.2444588244
b_2	2.578793	0.6304033399
b_3	3.497540	0.8549945951
b_4	2.507401	0.6129483581
b_5	1.265625	0.3093931973
a_1	1.0	0.2444588244
a_2	no coefficient	0.0000000000
a_3	no coefficient	0.0000000000
a_4	no coefficient	0.0000000000
a_5	no coefficient	0.0000000000

conversion process to see the results of a second-order cascade filter implemented in direct form.

For brevity, the phase plots are not shown. This FIR filter and the cascade filter will not exhibit linear phase characteristics. The emphasis of the output is in the magnitude plots. The issues to find out from the plots are how well the digital filter simulator tool performed in all these cases. The magnitude output plots do show the consistency of the software tool in handling the direct, cascade, and conversion processes. The outputs for the direct form, cascade form, and conversion process are all equivalent. Comparing these plots show the magnitude responses to be the same. This filter validates the software tool.

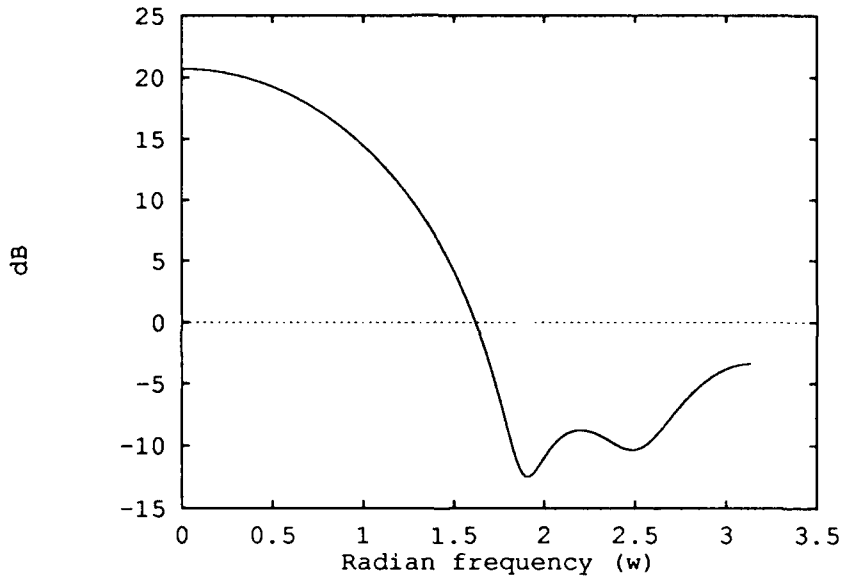


Figure 5.20. Single precision Magnitude Plot, conversion from cascade structure to direct structure, representation of coefficients in Table 5.7.

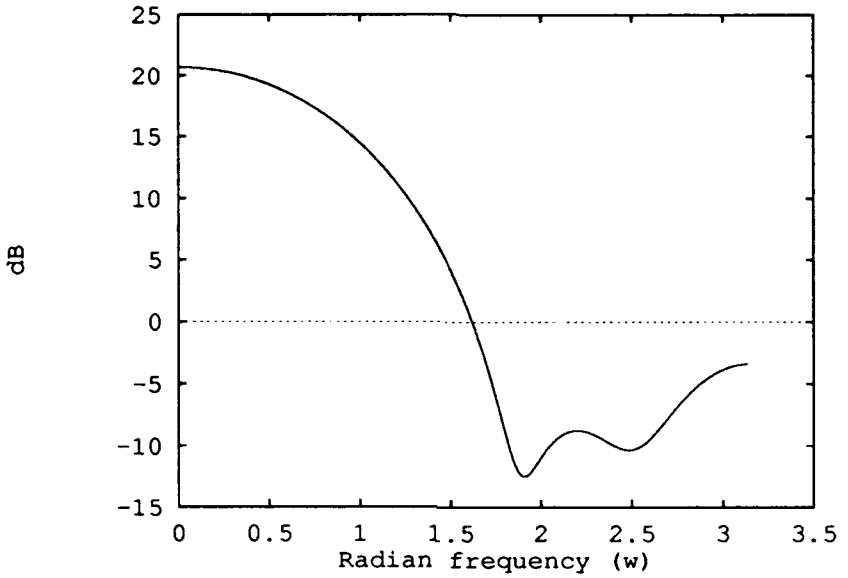


Figure 5.21. Quantized Magnitude Plot, direct structure 16-Bits precision, representation of coefficients in Table 5.5.

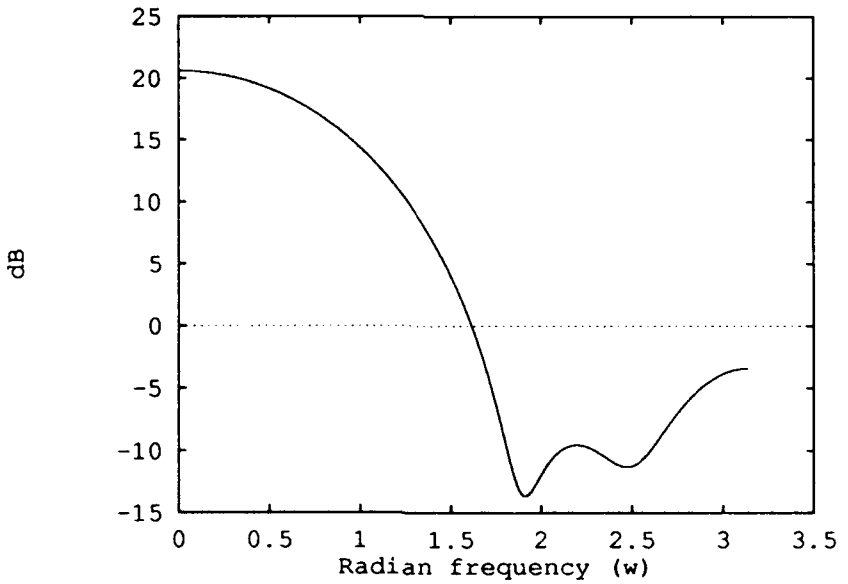


Figure 5.22. Quantized Magnitude Plot, direct structure 8-Bits precision, representation of coefficients in Table 5.5.

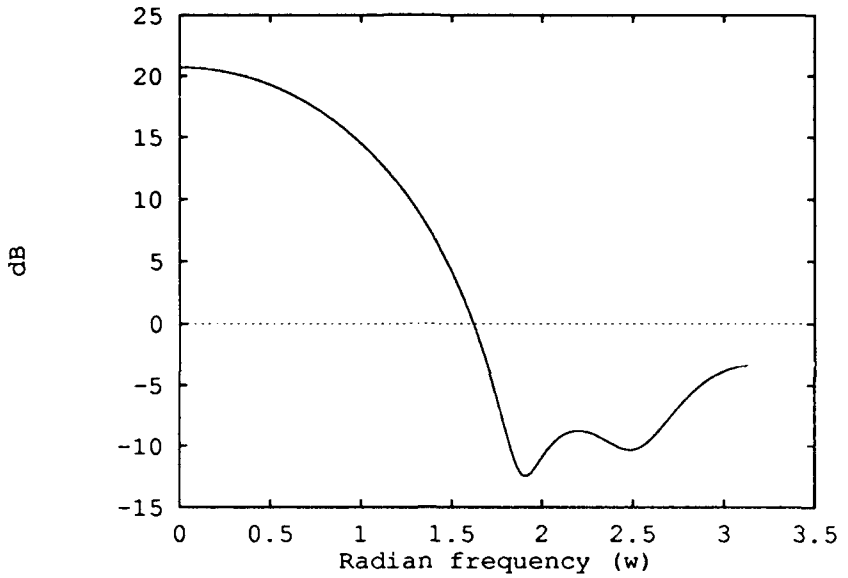


Figure 5.23. Quantized Magnitude Plot, cascade structure 16-Bits precision, representation of coefficients in Table 5.6.

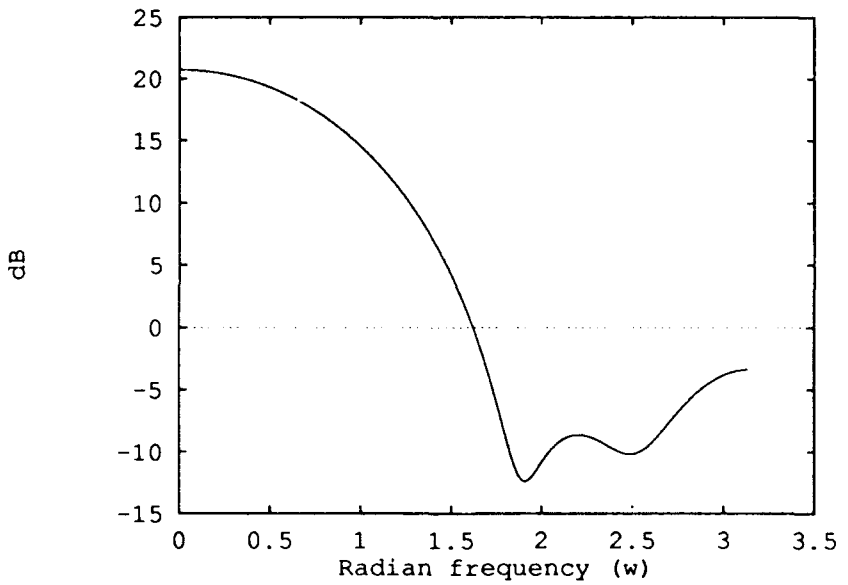


Figure 5.24. Quantized Magnitude Plot, cascade structure 8-Bits precision, representation of coefficients in Table 5.6.

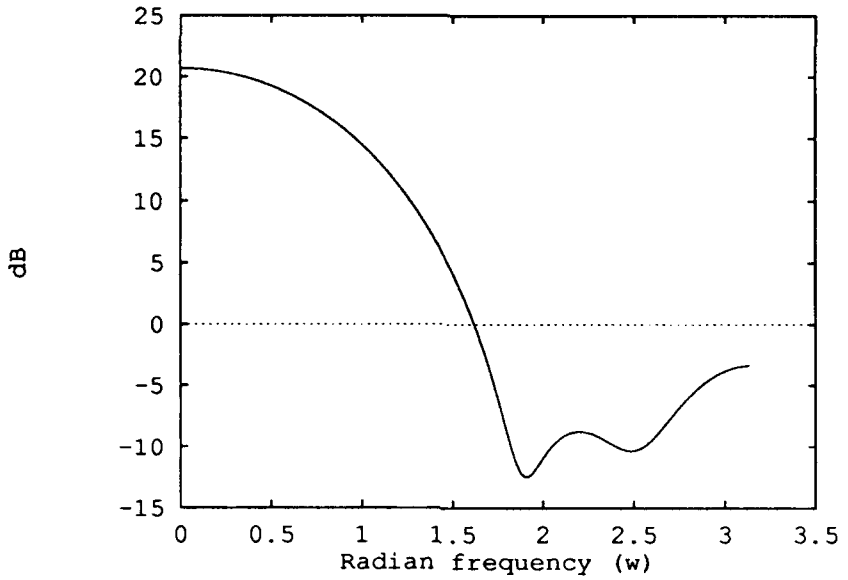


Figure 5.25. Magnitude Plot, conversion from cascade structure to direct structure using 16-bits precision, representation of coefficients in Table 5.7.

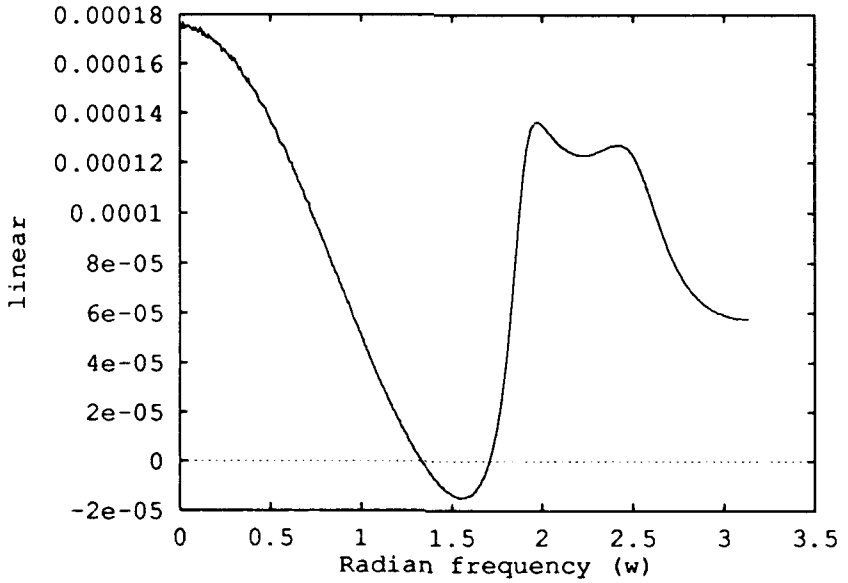


Figure 5.26. Linear Magnitude Error with direct structure, 16-Bits precision, representation of coefficients in Table 5.5.

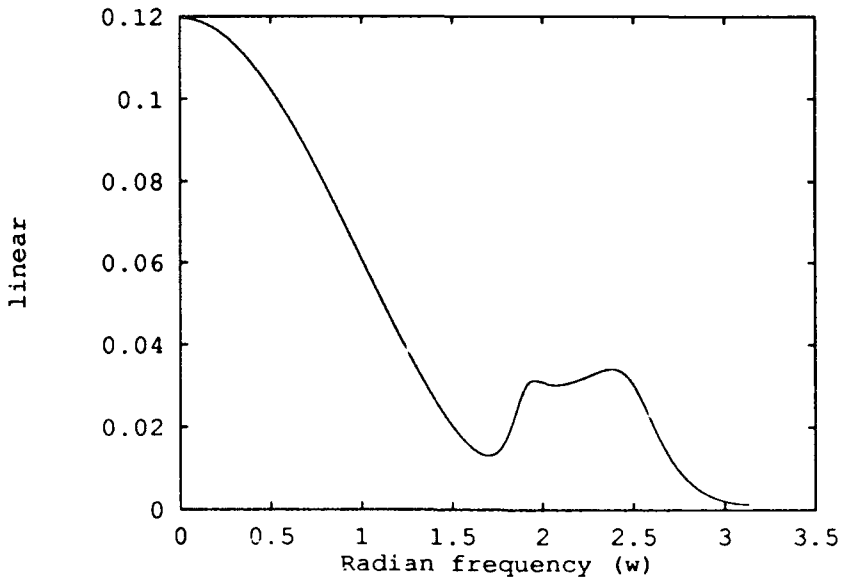


Figure 5.27. Linear Magnitude Error with direct structure, 8-Bits precision, representation of coefficients in Table 5.5.

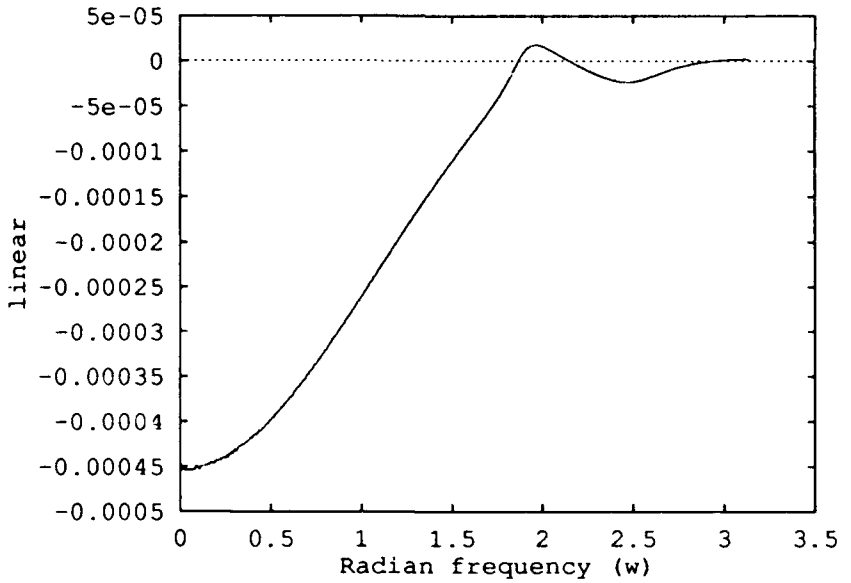


Figure 5.28. Linear Magnitude Error with cascade structure, 16-Bits precision, representation of coefficients in Table 5.6.

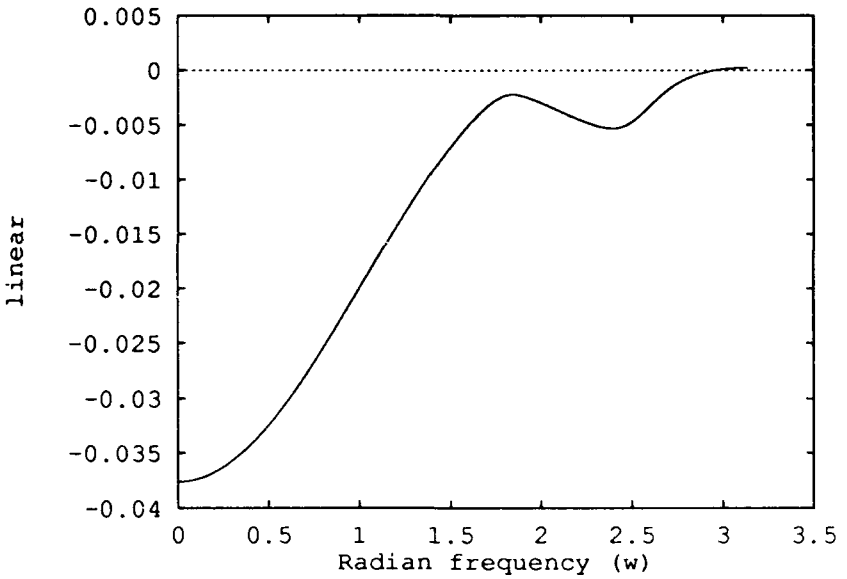


Figure 5.29. Linear Magnitude Error with cascade structure 8-Bits precision, representation of coefficients in Table 5.6.

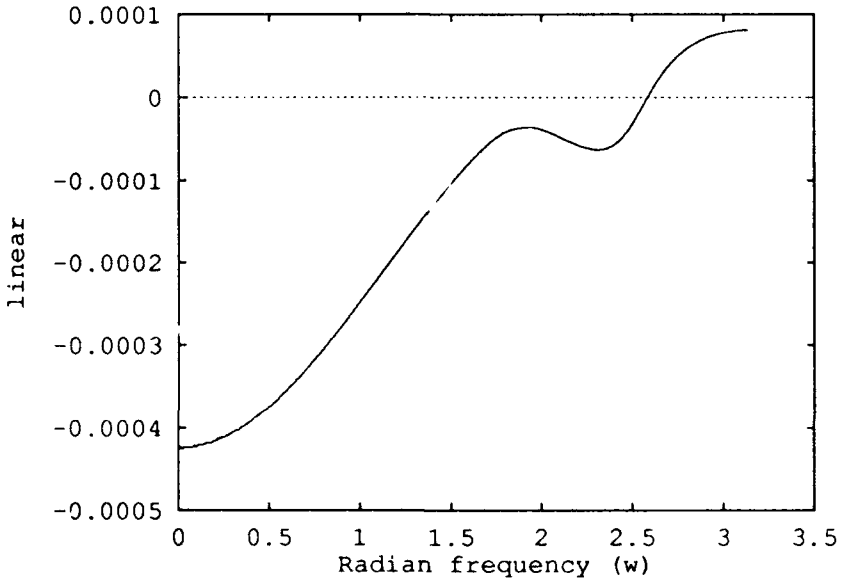


Figure 5.30. Linear Magnitude Error Plot, conversion process from cascade to direct, 16-bits precision, representation of coefficients in Table 5.7.

5.5 *IIR Bandpass Example Implemented in Direct Form to Show Robustness in IIR Filter*

The following filter is a band pass IIR filter that uses the direct structure. This example shows how some designs will maintain their form even when the precision to represent the numbers is reduced to only five bits to the right of the binary point.

The plots shown are using the 24-bit unquantized version, 16-bits, 12-bits, 8-bits, and 6-bits to represent the coefficients. The error plots show how much deviation from the 24-bit version is generated.

Since the phase for this filter is nearly linear in the pass band region, the phase plots are included. The phase plots will show that phase maintains linearity under the quantization process. Consider the following difference equation.

$$1.0y[n] + 0.0y[n-1] + 1.3805618y[n-2] + 0.0y[n-3] + 0.53138722y[n-4] = 0.03771x[n] + 0.0x[n-1] + (-0.07542x[n-2]) + 0.0x[n-3] + 0.03771x[n-4] \quad (5.5)$$

The coefficients used to generate the results is shown in Table 5.8 in both non-normalized and normalized representations.

Table 5.8. Coefficients for IIR Bandpass Example.

Coefficient Position	Coefficient	Normalized Coefficient
b_1	0.03771	0.0273149665
b_2	0.0	0.0000000000
b_3	-0.07542	-0.0546299331
b_4	0.0	0.0000000000
b_5	0.03771	0.0273149665
a_1	1.0	0.7243427634
a_2	0.0	0.0000000000
a_3	1.3805618	1.0000000000
a_4	0.0	0.0000000000
a_5	0.53138722	0.3849065006

The magnitude plots show that the response compared to the 24-bit version will stay within three percent of the frequency response even in Figure 5.39 using 6-bits to represent the coefficients. The error drops to 0.01% when 16-bits are used to represent the coefficients. This example shows how some filter designs can maintain similar frequency response with decreased accuracy for coefficient representation.

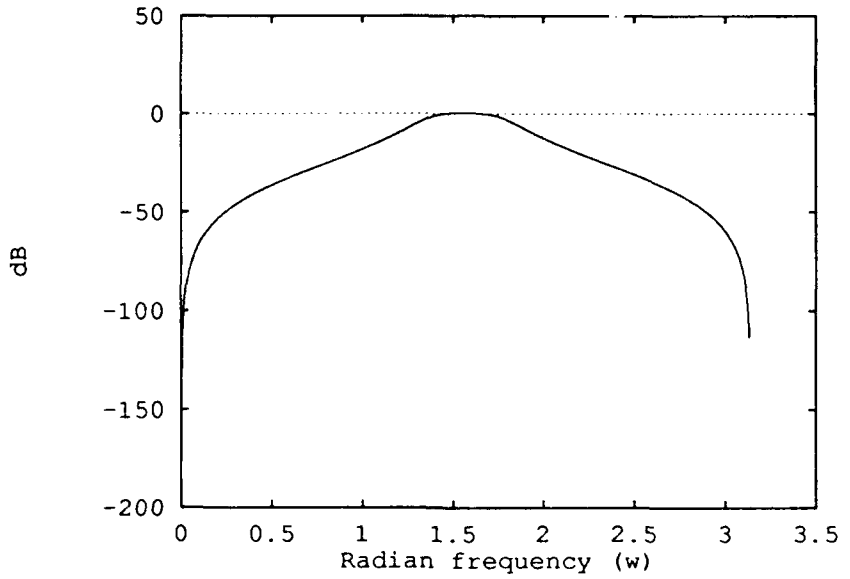


Figure 5.31. 24-bit Magnitude Plot, representation of coefficients in Table 5.8.

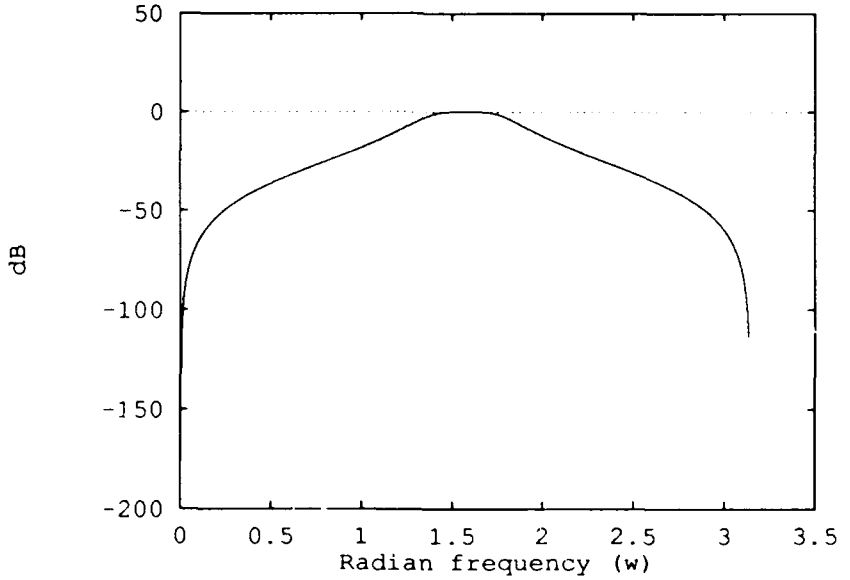


Figure 5.32. Quantized Magnitude Plot, 16-Bits precision, representation of coefficients in Table 5.8.

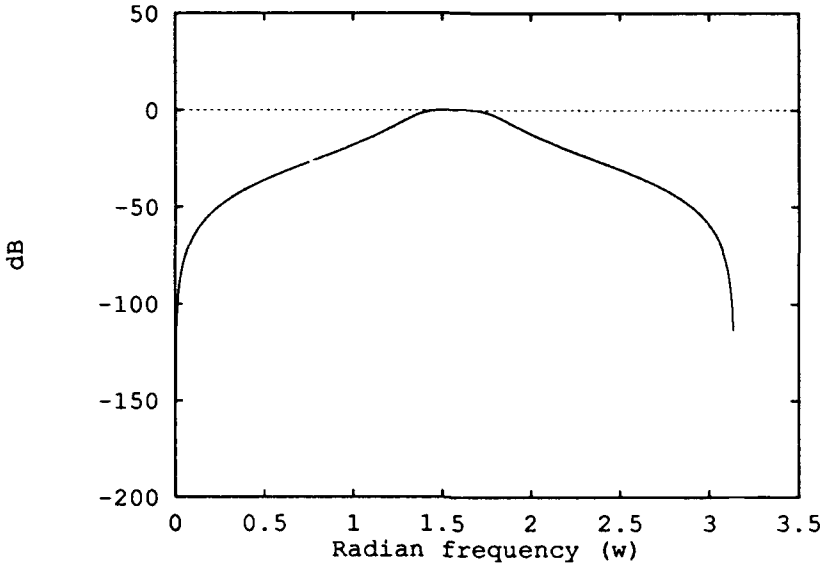


Figure 5.33. Quantized Magnitude Plot, 12-Bits resolution, representation of coefficients in Table 5.8.

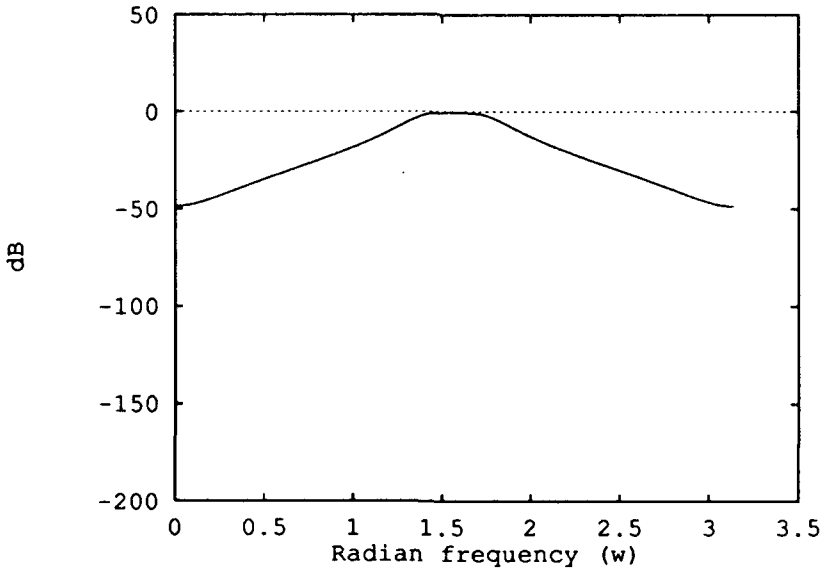


Figure 5.34. Quantized Magnitude Plot, 8-Bits resolution, representation of coefficients in Table 5.8.

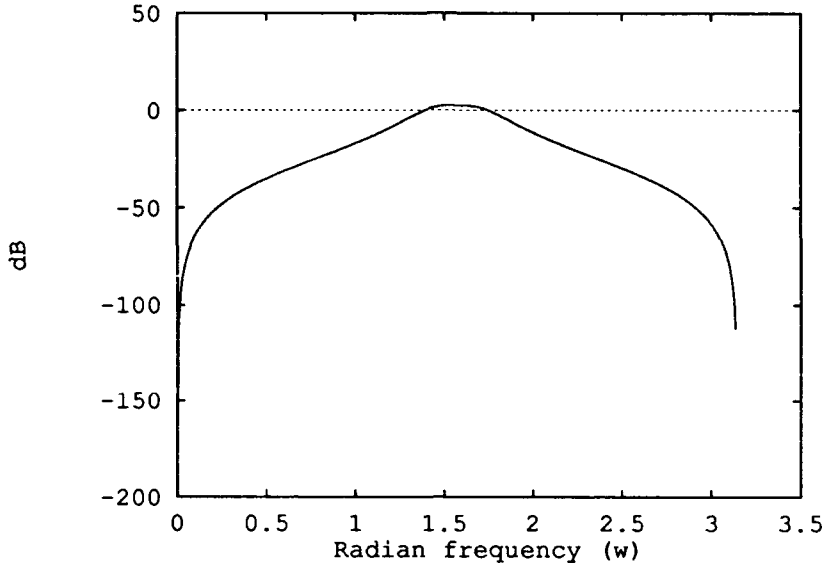


Figure 5.35. Quantized Magnitude Plot, 6-Bits resolution, representation of coefficients in Table 5.8.

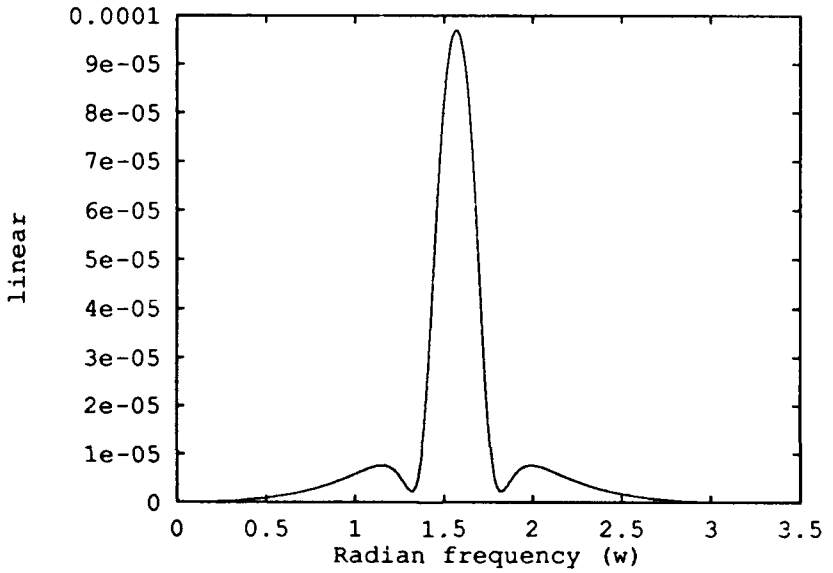


Figure 5.36. Linear Magnitude Error, 16-Bits resolution, representation of coefficients in Table 5.8.

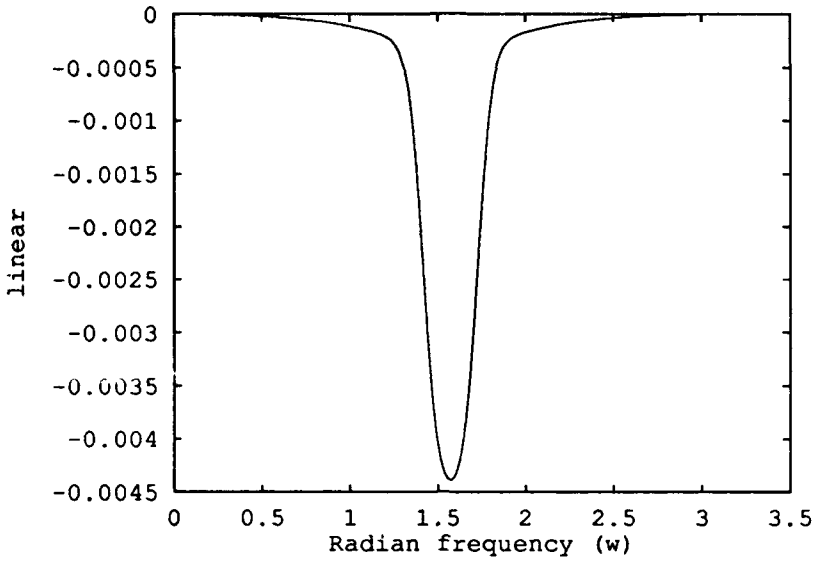


Figure 5.37. Linear Magnitude Error, 12-Bits resolution, representation of coefficients in Table 5.8.

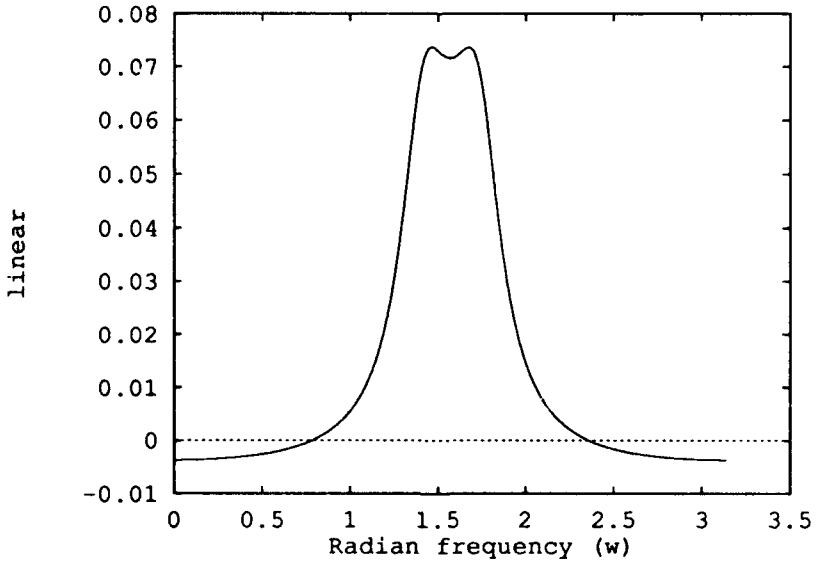


Figure 5.38. Linear Magnitude Error, 8-Bits resolution, representation of coefficients in Table 5.8.

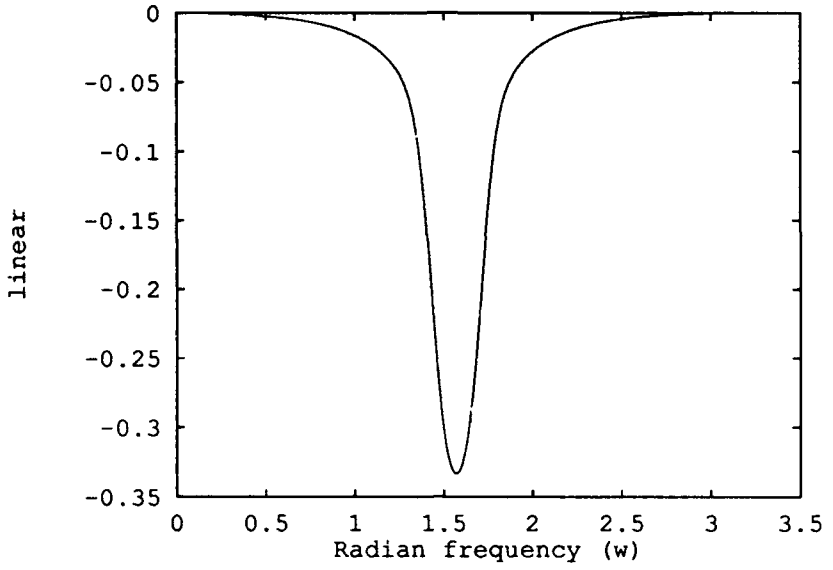


Figure 5.39. Linear Magnitude Error, 6-Bits resolution, representation of coefficients in Table 5.8.

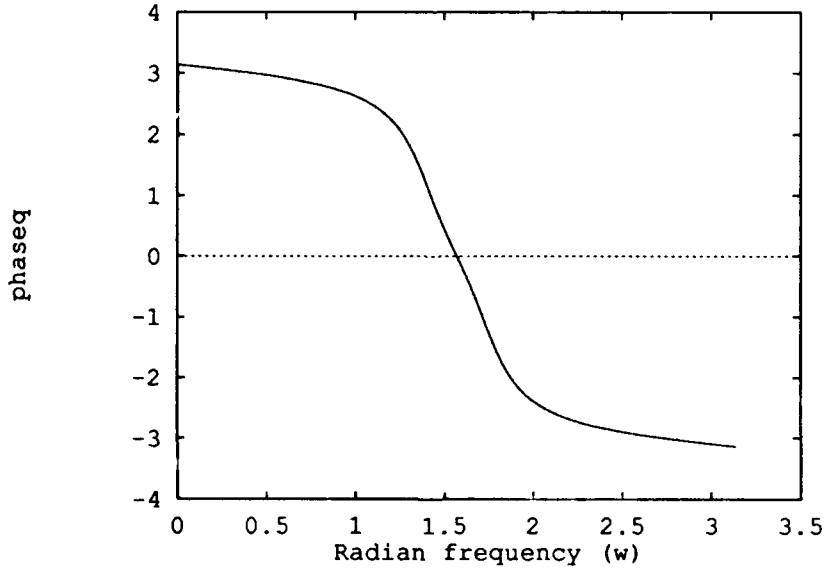


Figure 5.40. Phase Response, 16-Bits resolution, representation of coefficients in Table 5.8.

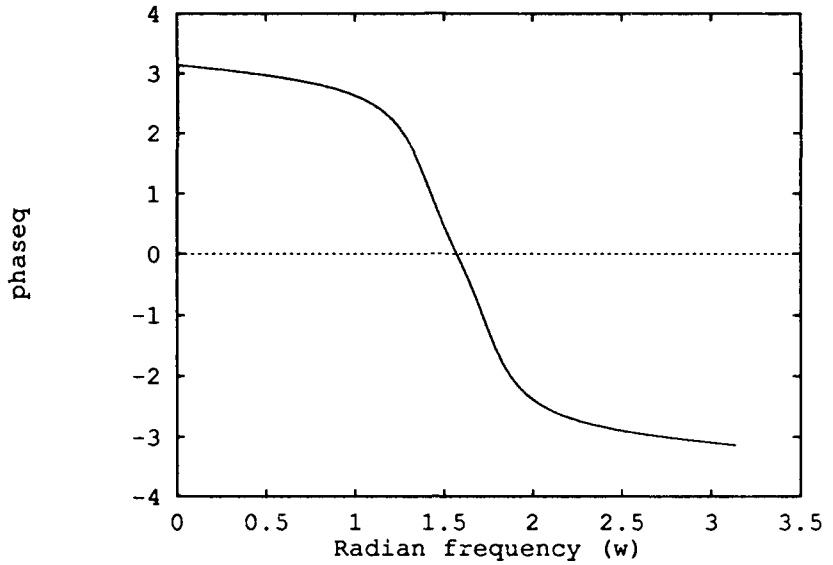


Figure 5.41. Phase Response, 12-Bits resolution,, representation of coefficients in Table 5.8.

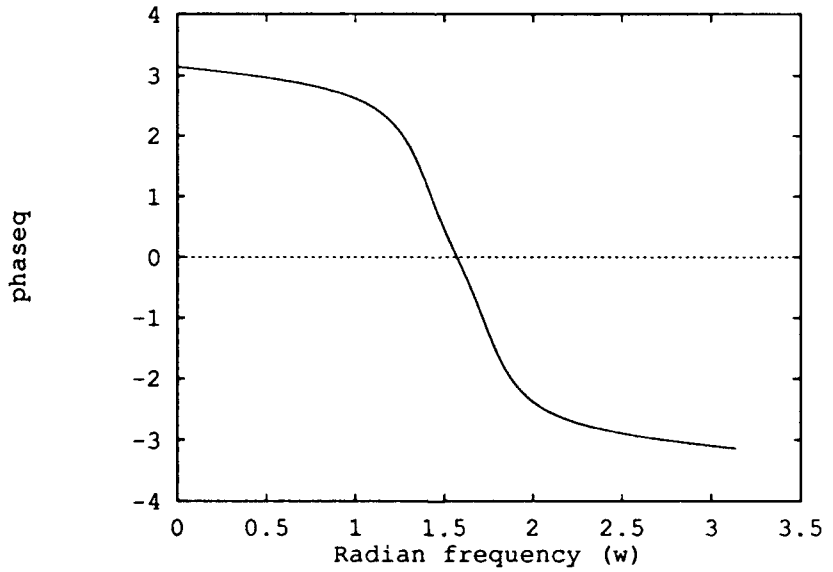


Figure 5.42. Phase Response, 8-Bits resolution, representation of coefficients in Table 5.8.

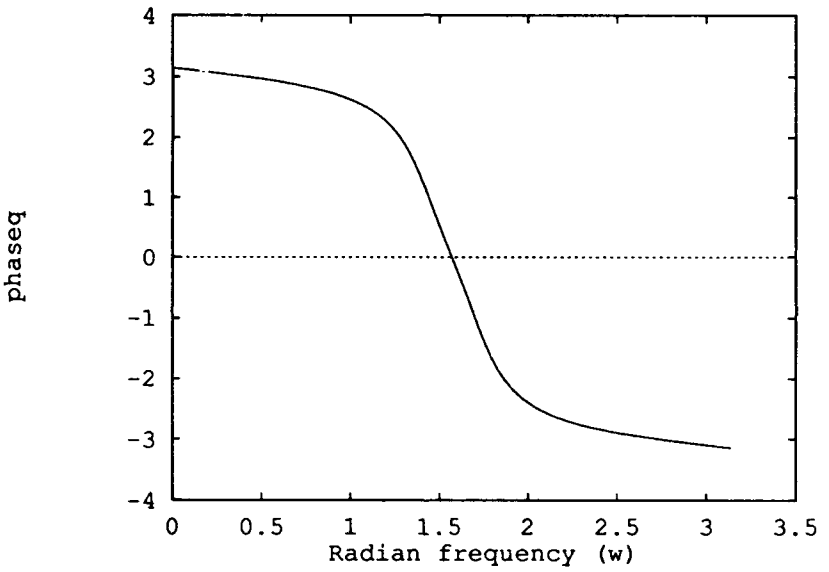


Figure 5.43. Phase Response, 6-Bits resolution, representation of coefficients in Table 5.8.

5.6 Cascade Bandpass Filter, Eighth-Order, Four Sections

This filter was taken from [2:454-557] and shows the pass band region changes shape during the quantization process. This example provides additional insight into the general shape of the magnitude plots when the number of bits used to represent the coefficients are reduced. The magnitude plots use 24-bits for full precision, 16-bits, 12-bits, and 8-bits.

The error plots show that the filter is very resilient to the quantization effects in the stop band regions. The errors in the magnitude response occur in the pass band region. Even when the use of eight bits for the coefficients is used, the difference in the magnitude plot occurs in the pass band region. With eight bits, there is a difference of 7dB in the pass band region.

One common characteristic of the cascade filter is their insensitivity to the effects of coefficient quantization. This is due to the limit of the power of z in the frequency equation.

The phase for this IIR cascade structure filter is non-linear. Therefore, the phase plots are omitted. The error plots show how much the magnitude plot is different from the 24-bit unquantized magnitude plot. Consider the following transfer function,

$$H(z) = \left(\frac{1.0 + 1.0z^{-1} + 0.0z^{-2}}{1.0 - (-1.609375)z^{-1} - 1.0z^{-2}}\right)\left(\frac{1.0 + (-.5937500)z^{-1} + 1.0z^{-2}}{1.0 - (-1.328125)z^{-1} - 1.00z^{-2}}\right) \left(\frac{1.0 + .890625z^{-1} + (-0.921875)z^{-2}}{1.0 - 1.046875z^{-1} - (-0.9375)z^{-2}}\right)\left(\frac{1.0 + 0.8125z^{-1} + (-0.96875)z^{-2}}{1.0 - 1.15625z^{-1} - (-0.984375)z^{-2}}\right). \quad (5.6)$$

The coefficients for this filter are shown in Table 5.9. This cascade filter has four second order sections. When the sections are multiplied, an eighth-order filter results.

Table 5.9. Unquantized (single precision) and quantized coefficients for an IIR Cascade filter.

Coefficient Position	Coefficient Value
Section Number One	
a_1	1.0
a_2	+.890625
a_3	-0.921875
b_1	1.0
b_2	0.0
b_3	1.0
Section Number Two	
a_1	1.0
a_2	1.046875
a_3	-0.9375
b_1	1.0
b_2	-1.609375
b_3	1.0
Section Number Three	
a_1	1.0
a_2	0.8125
a_3	-0.96875
b_1	1.0
b_2	-.5937500
b_3	1.0
Section Number Four	
a_1	1.0
a_2	1.15625
a_3	-0.984375
b_1	1.0
b_2	-1.328125
b_3	1.00

Figure 5.44 shows the filter magnitude response when no quantization effects are injected into the simulation. Figures 5.45 through 5.47 show the effects of reducing the number of bits to represent the coefficients from 16-bits to 8-bits. These plots show how the filter magnitude response degrades as the precision deceases. More importantly, the cascade filter shown in Table 5.9 withstands the degraded accuracy for the coefficients more so than compared to the filter in Section 5.4.

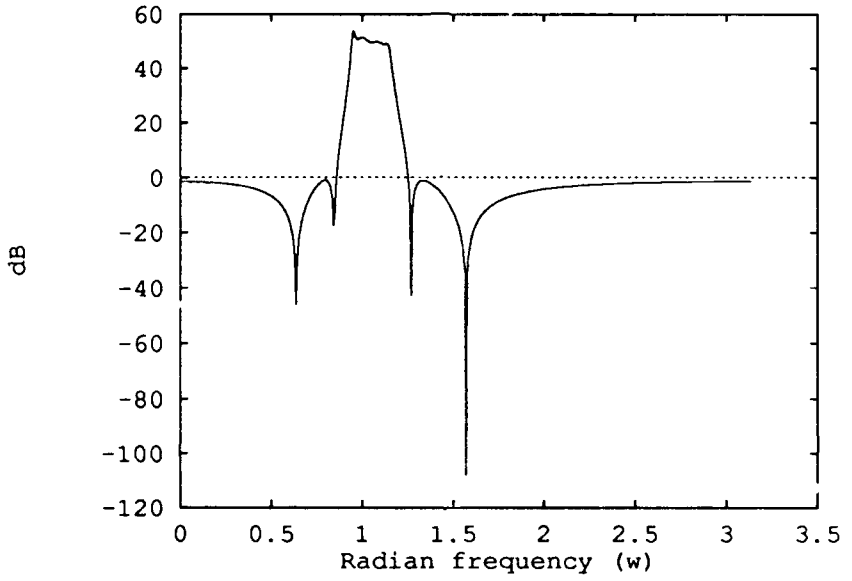


Figure 5.44. 24-bit single precision Magnitude Plot, cascade structure, representation of coefficients in Table 5.9.

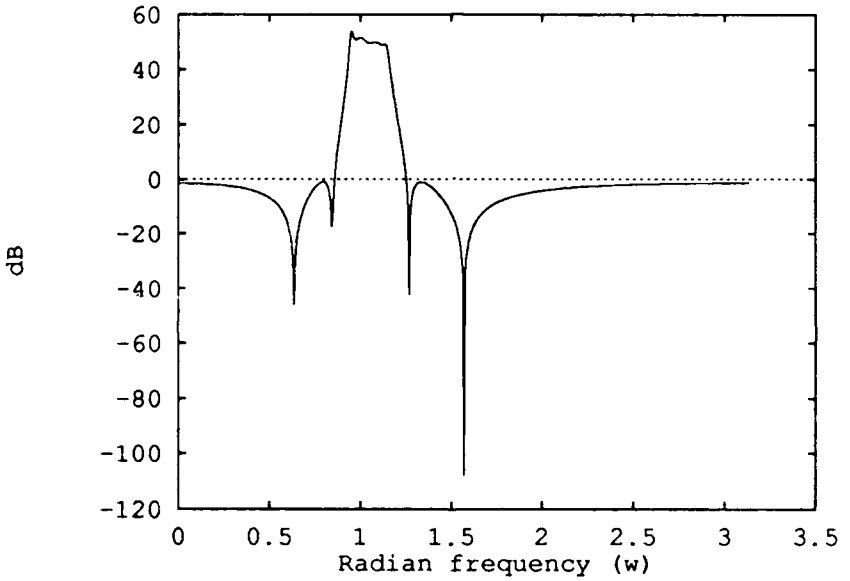


Figure 5.45. Quantized Magnitude Plot, 16-Bits precision, cascade structure, representation of coefficients in Table 5.9.

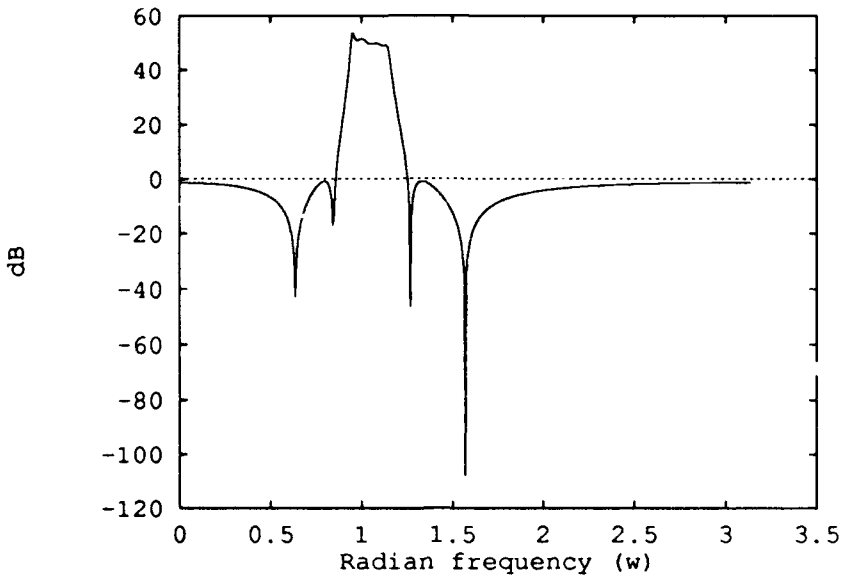


Figure 5.46. Quantized Magnitude Plot, 12-Bits precision, cascade structure, representation of coefficients in Table 5.9.

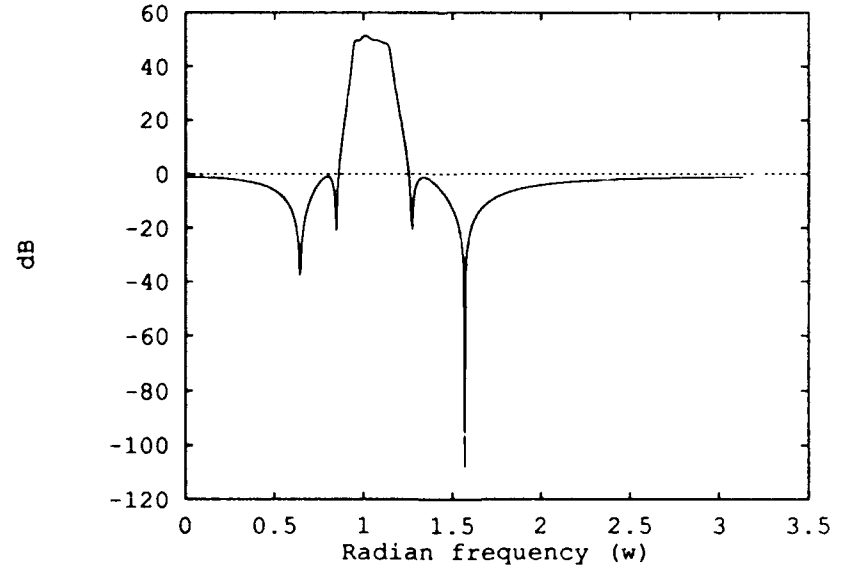


Figure 5.47. Quantized Magnitude Plot, 8-Bits precision, cascade structure, representation of coefficients in Table 5.9.

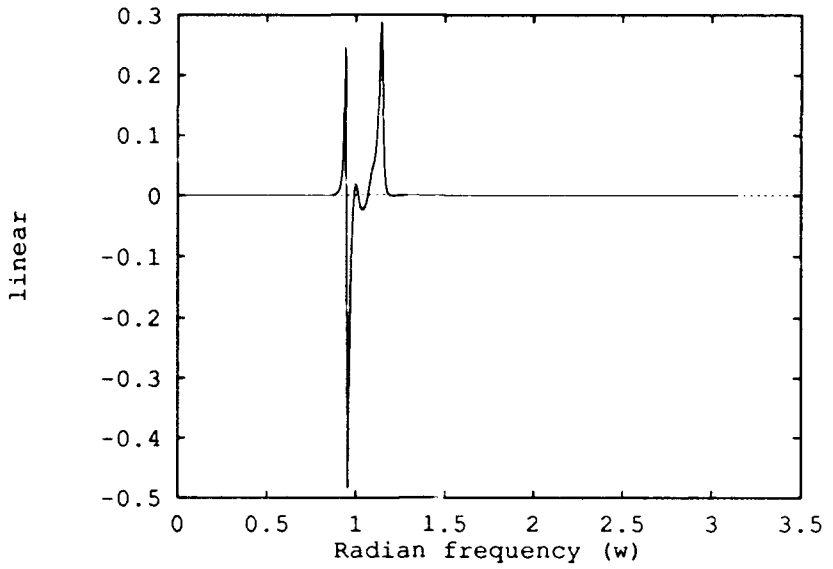


Figure 5.48. Linear Magnitude Error with 16-Bits for Coefficients, cascade structure, representation of coefficients in Table 5.9.

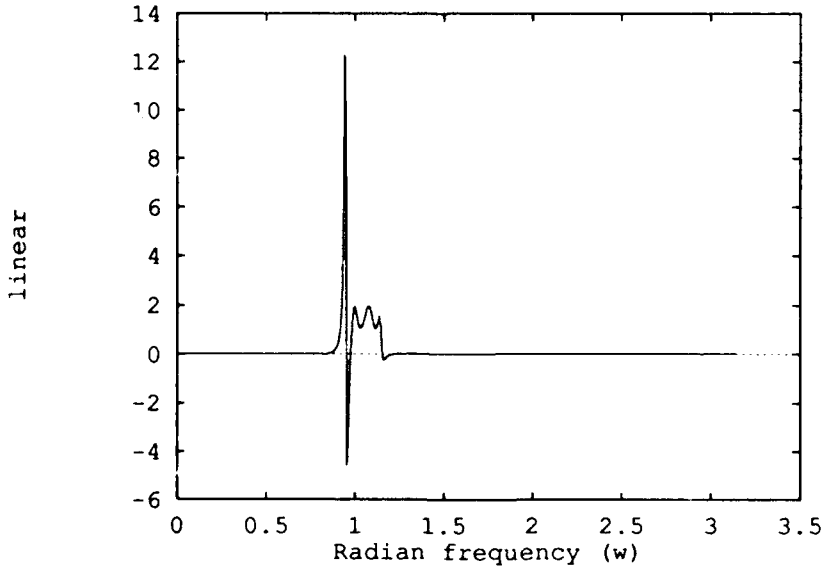


Figure 5.49. Linear Magnitude Error with 12-Bits for Coefficients, cascade structure, representation of coefficients in Table 5.9.

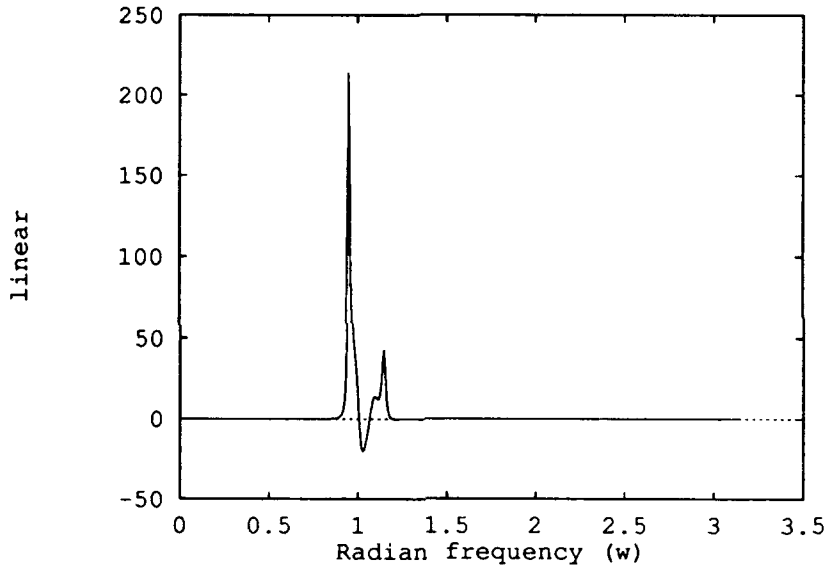


Figure 5.50. Linear Magnitude Error with 8-Bits for Coefficients, Cascade structure, representation of coefficients in Table 5.9.

5.7 Cascade Filter, Twelfth-Order

The advantages of the cascade design becomes most obvious when the performance requires a large difference between the pass band and the stop band. The filter shown in this section is a band pass IIR elliptic filter [15:338-341]. The filter was designed with the following performance specifications:

$$\begin{aligned} 0.99 \leq &| H(e^{j\omega}) | \leq 1.01, && 0.3\pi \leq \omega \leq 0.4\pi, \\ &| H(e^{j\omega}) | \leq 1.01(-40dB), && \omega \leq 0.29\pi, \\ &| H(e^{j\omega}) | \leq 1.01(-40dB), && 0.41\pi \leq \omega \leq \pi. \end{aligned}$$

The coefficients in Table 5.10 show the need for the normalization procedure. Table 5.10 shows the coefficients from the design process in column two in single precision (unquantized). Column three quantizes the coefficients to 16-bits of precision with the decimal point to the left of the left most bit. Three coefficients are truncated in this case because they are greater then the magnitude of unity. When a coefficient value is greater than the magnitude of unity, the quantized value will have additional quantization error than just from the constraint of the word register length. These coefficients are out of range for the quantizer routine. For example, in column three, section four, coefficient position a_2 the coefficient value meets the out of range condition. Column four normalizes the coefficients from column one first. Then, after normalization, the coefficients are represented using two's complement with 16-bits precision.

After the normalization routine is run, the values of the coefficients are all changed. This will not effect the filter's performance, since the normalization applies a gain universally to all the coefficients. The gain factor can easily be found by finding the gain applied to the coefficient that was normalized to unity.

When a user starts with a set of coefficients for the cascade structure, many possible states can exist for the coefficient file. This example shows all possibilities that can be generated by the digital filter software tool.

Only one coefficient file is entered into the digital filter simulator tool. The rest of the coefficient files are computed by the digital filter simulator tool. Each coefficient file generated can be out of range for the quantizer, so the normalization routine will be needed to create another coefficient file. It may seem that there are more files to keep track of, but only the normalized coefficient files are needed to see the performance results.

Table 5.10. Unquantized and quantized coefficients for an IIR Cascade filter.

Coefficient Position	Coefficient Value	Quantized To 16-bits	Normalized And Quantized To 16-bits
Section Number One			
a_1	1.	1.000000	0.852447
a_2	0.738409	.738403	0.629455
a_3	-0.850835	-0.850830	-0.725311
b_1	0.135843	0.135833	0.115814
b_2	0.026265	0.026275	0.022399
b_3	0.135843	0.135833	0.115814
Section Number Two			
a_1	1.	1.000000	0.852447
a_2	0.960374	0.960388	0.818664
a_3	-0.860000	-0.859985	-0.733123
b_1	0.278901	0.278900	0.237762
b_2	-0.444500	-0.444488	-0.378906
b_3	0.278901	0.278900	0.237762
Section Number Three			
a_1	1.	1.000000	0.852447
a_2	0.629449	0.629455	0.536590
a_3	-0.931460	-0.931457	-0.794036
b_1	0.535773	0.535766	0.456726
b_2	-0.249249	-0.249237	-0.212463
b_3	0.535773	0.535766	0.456726
Section Number Four			
a_1	1.	1.000000	0.852447
a_2	1.116458	1.000000	0.951721
a_3	-0.940429	-0.940429	-0.801666
b_1	0.697447	0.697448	0.594543
b_2	-0.891543	-0.891540	-0.760009
b_3	0.697447	0.697448	0.594543
Section Number Five			
a_1	1.	1.000000	0.852447
a_2	0.605182	0.605194	0.515899
a_3	-0.983693	-0.983703	-0.838562
b_1	0.773093	0.773101	0.6590270
b_2	-0.425920	-0.425933	-0.363067
b_3	0.773093	0.773101	.659027
Section Number Six			
a_1	1.	1.000	0.852447
a_2	1.173078	1.000	1.000000
a_3	-0.986166	-0.986175	-0.840667
b_1	0.917937	0.917938	0.782501
b_2	-1.122226	-1.000	-0.956665
b_3	0.917937	0.917938	0.782501

A user can choose the conversion utility to change the structure of the filter into a direct form. The question that arises is what coefficient file to use as input for the conversion utility. The user is faced with using the non-normalized coefficient file or with using the normalized coefficient file. The conversion utility, however, takes the current coefficient file (coefficients for cascade structure) shown on screen and computes a new set of coefficients for implementation using direct structure. The results from using non-normalized coefficients or normalized coefficients for the conversion process is found in Table 5.12. Either path to generate the direct structure will result in the same set of coefficients as shown in Table 5.12.

The magnitude of the coefficients will affect the resulting quantized values. When a coefficient is smaller than the smallest increment of the quantizer, a value of zero results for the output. Even with using only six bits, the quantizer translates a filter coefficient into the number zero when the magnitude is less than $1/2^5$.

Once the conversion utility is run, two cases will result. The first case is where the magnitude of a converted coefficient exceeds unity. The second case is where the magnitude of all converted coefficients are less than unity.

There is, however, a secondary section in the normalization routine. The user will have the option, when running the normalization routine, to normalize the coefficients even when the magnitude of the coefficients are less than one. This process will maximize the dynamic range by setting the largest coefficient to a magnitude of 1.0.

Table 5.11 shows the results found by taking the coefficients from the cascade structure and applying the normalization routine to the coefficients. The normalized coefficients are used to build the output plots for the cascade structure.

Table 5.12 shows the resulting coefficient file after the conversion and normalization utilities are run. These coefficients now refer to a direct structure (the results of the conversion utility is the direct structure). It's easily seen that the order of the filter is still a 12th order filter. The poles and the zeros are still in the same location, but the structure has changed to implement the poles and zeros.

The coefficients from Table 5.11 column two (the cascade structure, normalized coefficients) are used for the following set of graphs. Plots are provided to show how the cascade structure resists the effects of the quantization process. The magnitude plots and the phase plots are shown. The cascade filter does exhibit linear phase characteristics within the pass band region. In each case using the cascade filter, the design specifications were met.

Plots included in this example are in the following order:

Table 5.11. Coefficients for the Cascade structure and the results of the normalization.

Coefficient	Coefficient Normalized (Unquantized)
Section Number One	
1.	0.8524582
0.738409	0.6294628
-0.850835	-0.7253013
0.135843	0.1158004
0.026265	0.0223898
0.135843	0.1158004
Section Number Two	
1.	0.8524582
0.960374	0.8186787
-0.860000	-0.7331141
0.278901	0.2377514
-0.444500	-0.3789176
0.278901	0.2377514
Section Number Three	
1.	0.8524582
0.629449	0.5365790
-0.931460	-0.7940307
0.535773	0.4567241
-0.249249	-0.2124743
0.535773	0.4567241
Section Number Four	
1.	0.8524582
1.116458	0.9517338
-0.940429	-0.8016764
0.697447	0.5945444
-0.891543	-0.7600032
0.697447	0.5945444
Section Number Five	
1.	0.8524582
0.605182	0.5158923
-0.983693	-0.8385572
0.773093	0.6590295
-0.425920	-0.3630790
0.773093	0.6590295
Section Number Six	
1.	0.8524582
1.173078	1.0000000
-0.986166	-0.8406653
0.917937	0.7825029
-1.122226	-0.9566508
0.917937	0.7825029

Table 5.12. Results of the Conversion process from Non-Normalized coefficients and Normalized coefficients; the resulting direct structure's coefficients are then normalized.

Coefficients From the Conversion Process for Direct Structure		
Coefficient	Conversion From Non-Normalized Coefficients	Conversion From Normalized Coefficients
a_1	0.0113164	0.0113164
a_2	-0.0591053	-0.0591053
a_3	0.1897304	0.1897305
a_4	-0.4166786	-0.4166786
a_5	0.7045002	0.7045002
a_6	-0.9326622	-0.9326623
a_7	1.0000000	1.0000000
a_8	-0.8618795	-0.8618796
a_9	0.6015846	0.6015846
a_{10}	-0.3287177	-0.3287176
a_{11}	0.1382653	0.1382653
a_{12}	-0.0397719	-0.0397719
a_{13}	0.0070364	0.0070364
b_1	0.0001136	0.0001136
b_2	-0.0005590	-0.0005590
b_3	0.0017028	0.0017028
b_4	-0.0036198	-0.0036198
b_5	0.0060357	0.0060357
b_6	-0.0080497	-0.0080497
b_7	0.0088791	0.0088791
b_8	-0.0080497	-0.0080497
b_9	0.0060357	0.0060357
b_{10}	-0.0036198	-0.0036198
b_{11}	0.0017028	0.0017028
b_{12}	-0.0005590	-0.0005590
b_{13}	0.0001136	0.0001136

1. Figure 5.51, 24-bit single precision Magnitude Plot, Cascade Structure.
2. Figure 5.52, 24-bit single precision Magnitude Plot, Highlighting the Passband.
3. Figure 5.53, Magnitude Plot, 16-bit Coefficients.
4. Figure 5.54, Magnitude Plot, 16-bits, Highlighting the Passband.
5. Figure 5.55, Magnitude Plot, 8-bit Coefficients.
6. Figure 5.56, Magnitude Plot, 8-bit Coefficients, Highlighting the Passband.
7. Figure 5.57, 24-bit Phase Plot, Normalized Coefficients.
8. Figure 5.58, 24-bit Phase Plot, Highlighting the Passband.
9. Figure 5.59, Phase Plot with 16-bit Coefficients.
10. Figure 5.60, Phase Plot with 16-bit Coefficients, Highlighting the Passband.
11. Figure 5.61, Phase Plot with 8-bit Coefficients.
12. Figure 5.62, Phase Plot with 8-bit Coefficients, Highlighting the Passband.

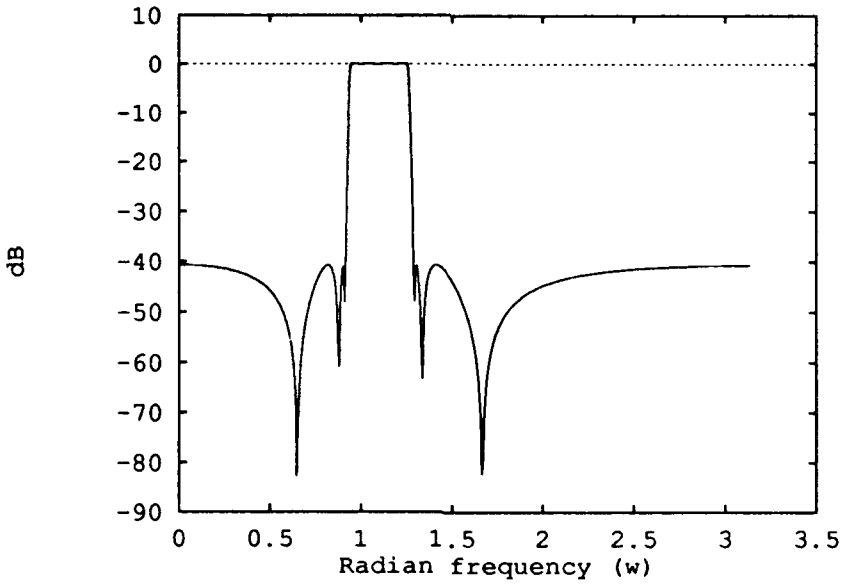


Figure 5.51. 24-bit Magnitude Plot, cascade structure representing coefficients in Table 5.11, column two.

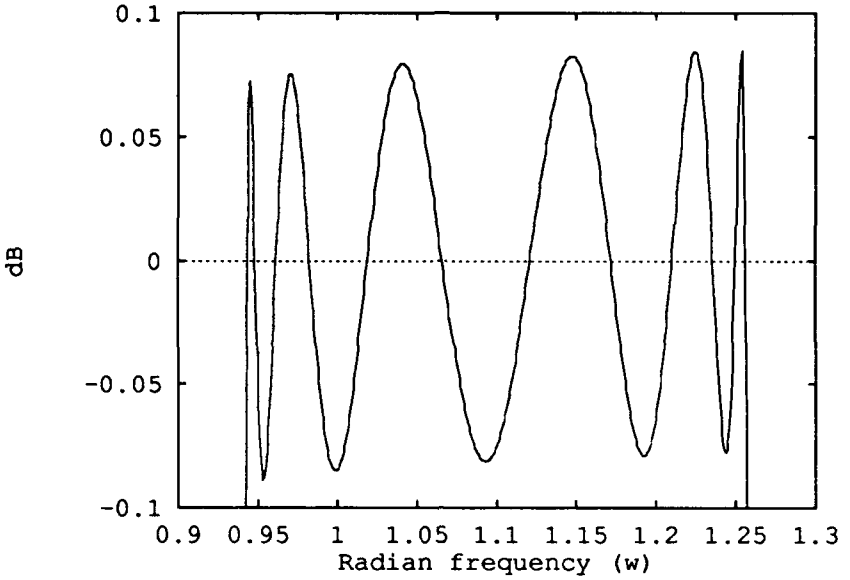


Figure 5.52. 24-bit Magnitude Plot, cascade structure representing coefficients in Table 5.11, column two.

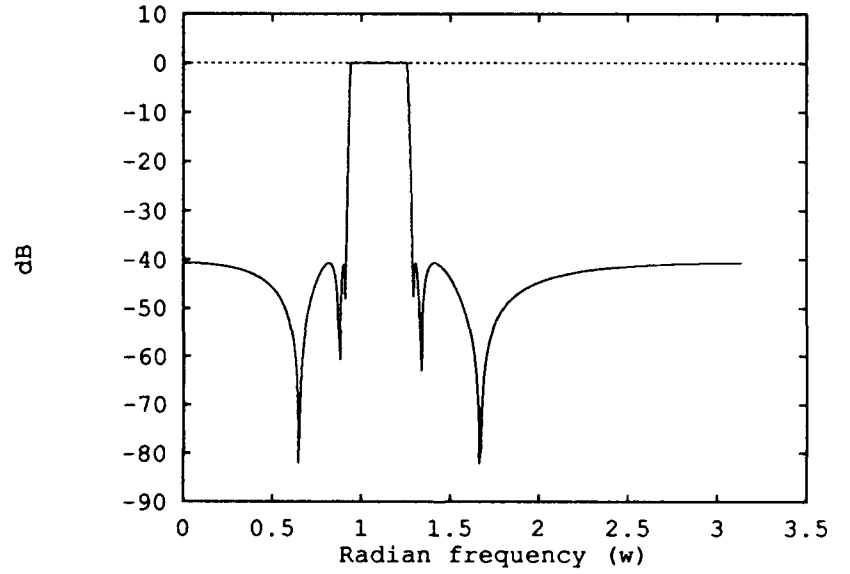


Figure 5.53. Magnitude Plot, 16-Bits normalized cascade representing coefficients in Table 5.10

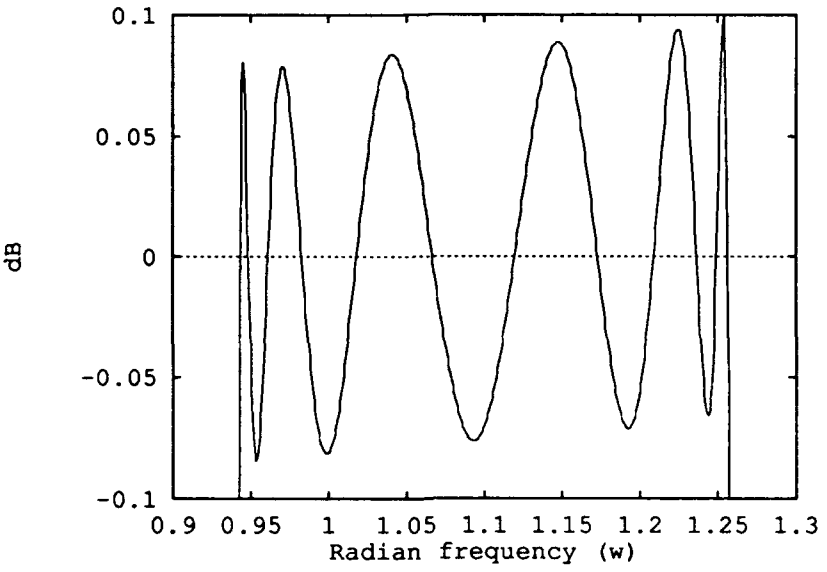


Figure 5.54. Magnitude Plot, 16-Bits normalized cascade representing coefficients in Table 5.10.

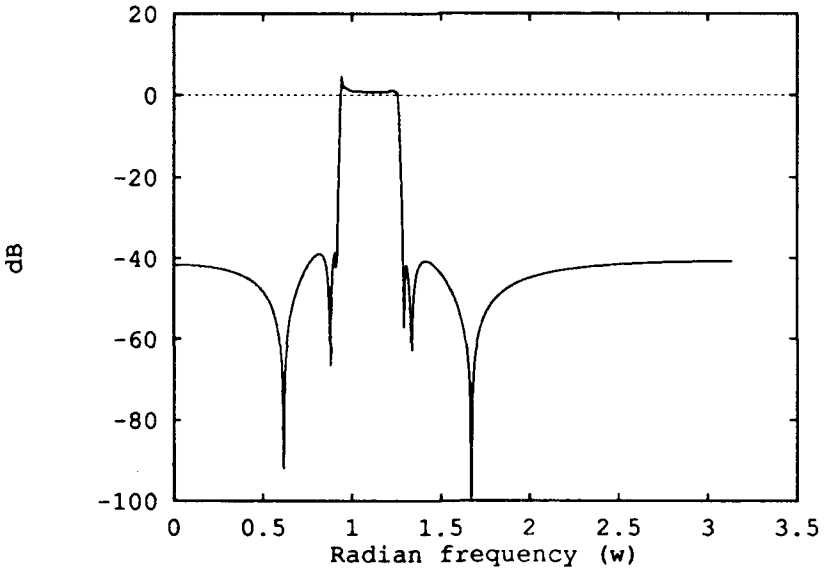


Figure 5.55. Magnitude Plot, 8-Bits normalized cascade representing coefficients in Table 5.11.

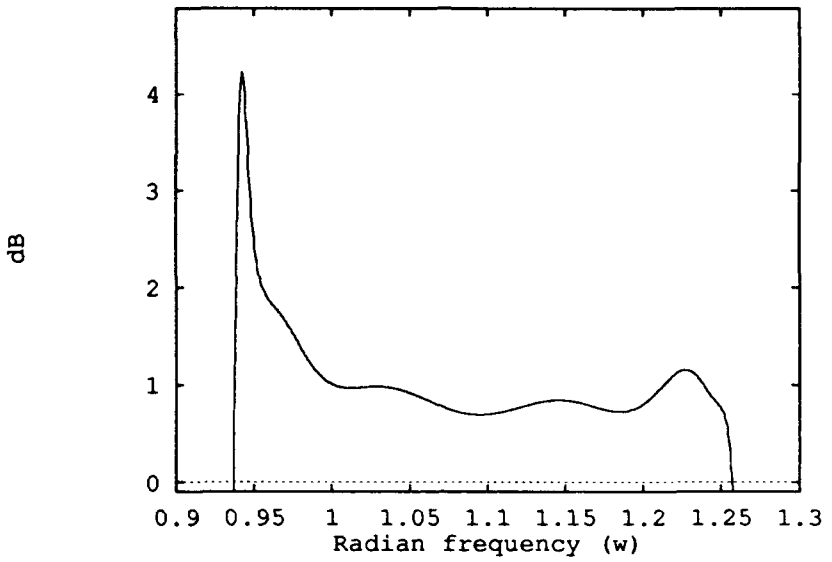


Figure 5.56. Magnitude Plot, 8-Bits normalized cascade representing coefficients in Table 5.11.

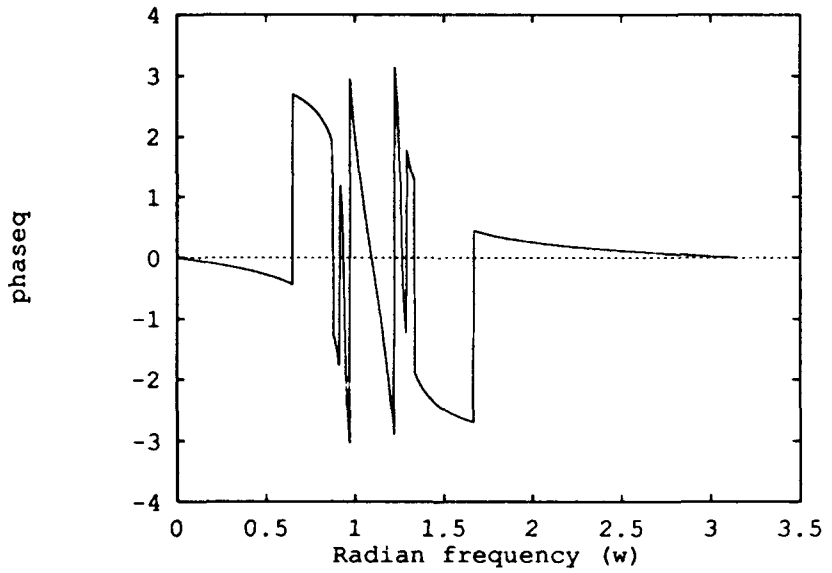


Figure 5.57. Phase Response 24-bit Coefficients, cascade structure representing coefficients in Table 5.11.

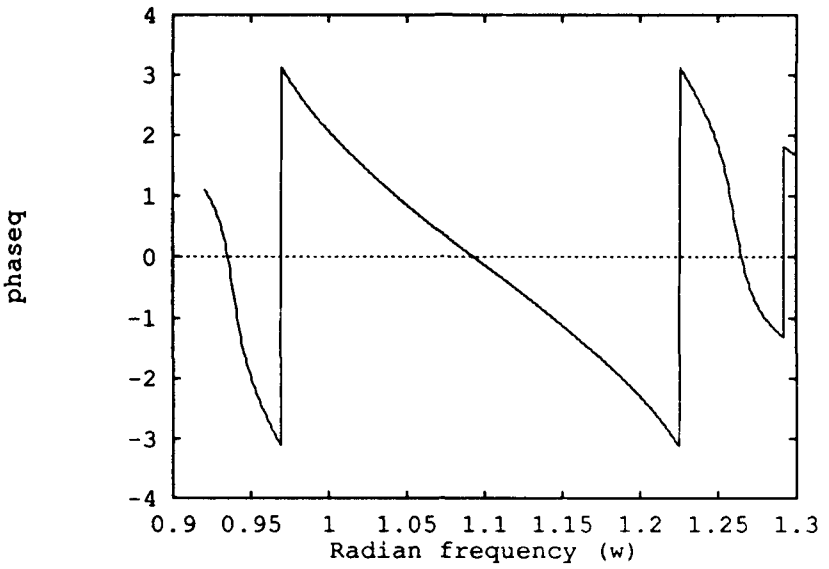


Figure 5.58. Phase Response 24-bit Coefficients, cascade structure representing coefficients in Table 5.11.

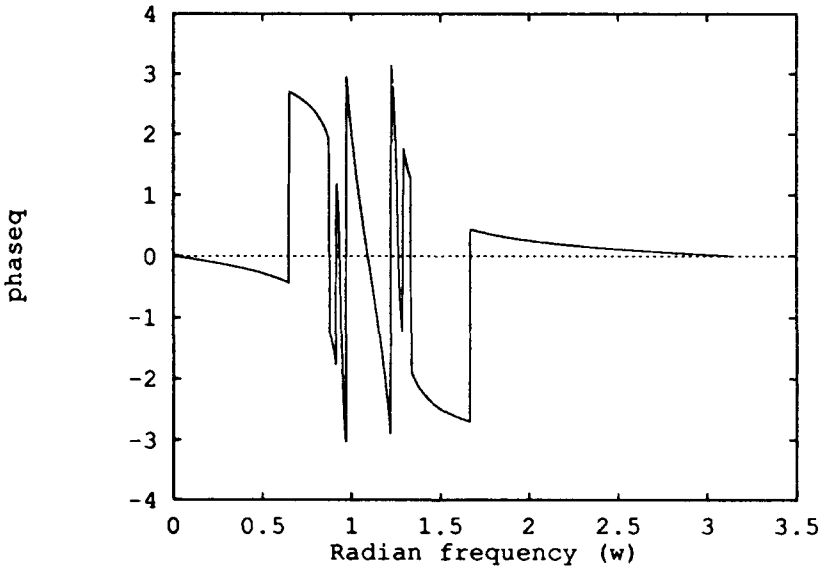


Figure 5.59. Phase Response with 16-Bits for Coefficients, cascade structure representing coefficients in Table 5.10.

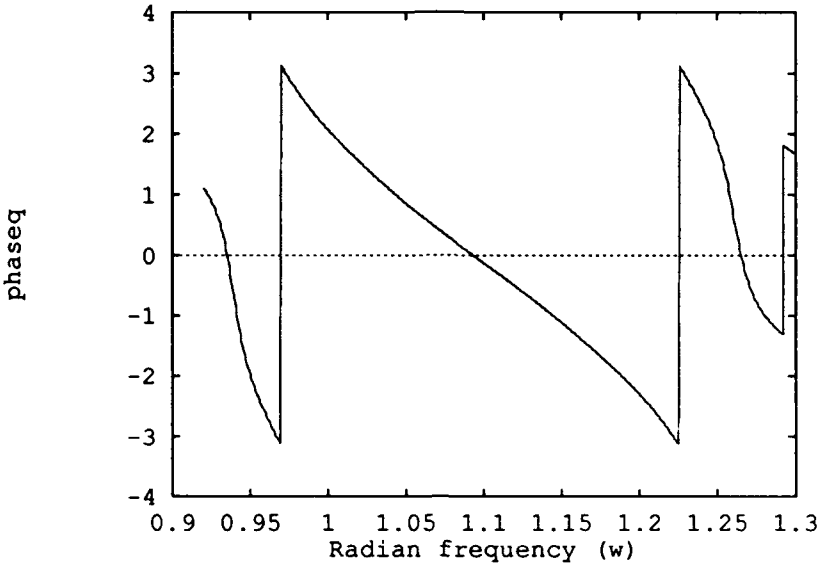


Figure 5.60. Phase Response with 16-Bits for Coefficients, cascade structure representing coefficients in Table 5.10.

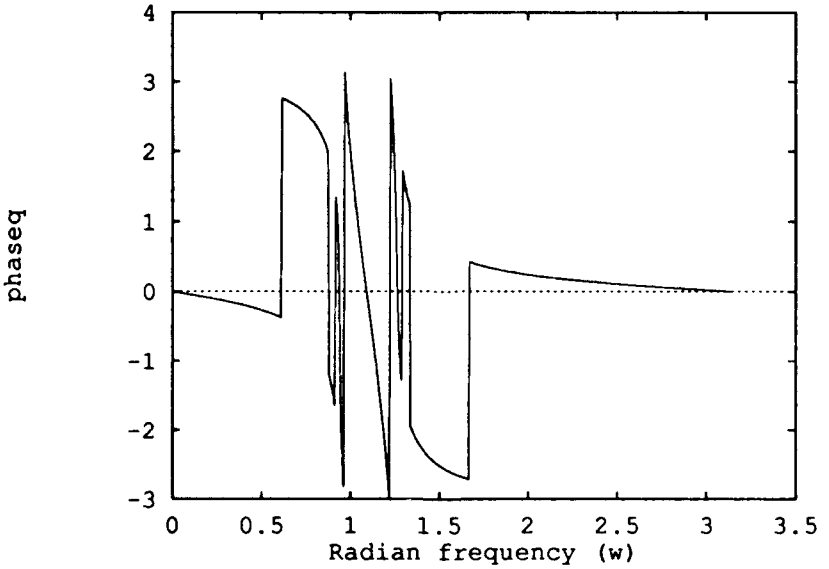


Figure 5.61. Phase Response with 8-Bits for Coefficients, cascade structure representing coefficients in Table 5.11.

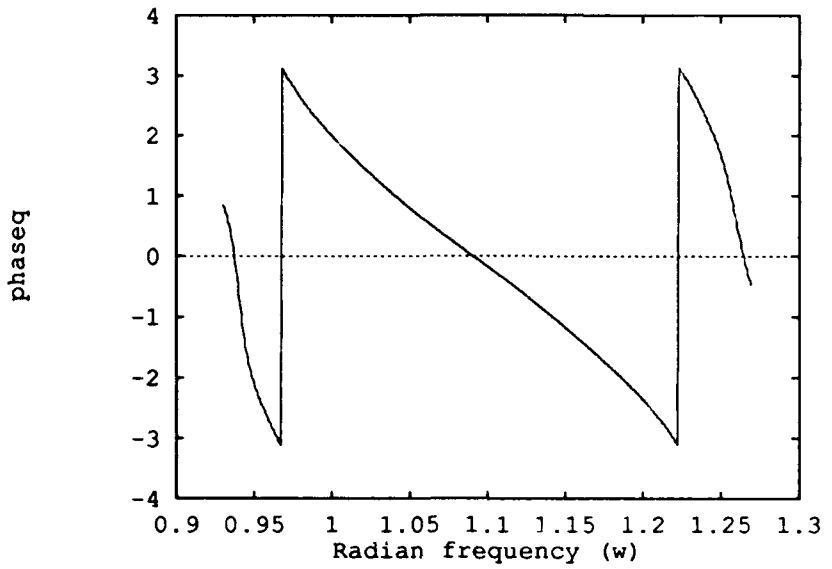


Figure 5.62. Phase Response with 8-Bits for Coefficients, cascade structure representing coefficients in Table 5.11.

In order to show the robustness of the cascade design, the plots for the conversion process are shown. The sensitivity of the filter to the effects of the quantization increases as the order of the filter increases. This is dramatically shown in this example. The quantization causes a shift in the locations of the poles and zeros. When this shift is subjected to the higher power multiplies in frequency, the change in frequency response correspondingly increased. That's why normally, the direct form is used for filters of second order or less. When higher order filters are needed, the cascade structure is chosen to maintain robustness in design.

The magnitude response with 8-bits is severely distorted. The basic idea is for the designer is to continue to add more and more bits to the word length until the design specifications can be met. When the design specifications are met, then the designer has attained an acceptable word length for the coefficients. The magnitude responses will continue to improve as more bits are added to increase the precision.

To emphasize the advantage of cascade structure over the use of the direct form, the filter was simulated with double precision 48-bit coefficients. The 48-bit magnitude plot uses double precision representation. Notice that the 48-bit magnitude response continues to approach the magnitude plots produced by the cascade structure. The direct structure can only approach the desired response even with the use of 48-bit coefficients.

The list of plots that follows are:

1. Magnitude Plot, 8-bits, normalized conversion from the cascade structure, direct form I.
2. Magnitude Plot, 16-bits, normalized conversion from the cascade structure, direct form I.
3. Magnitude Plot, 24-bits, normalized conversion from the cascade structure, direct form I.
4. Magnitude Plot, Un-quantized double precision, normalized conversion from the cascade structure, direct form I.

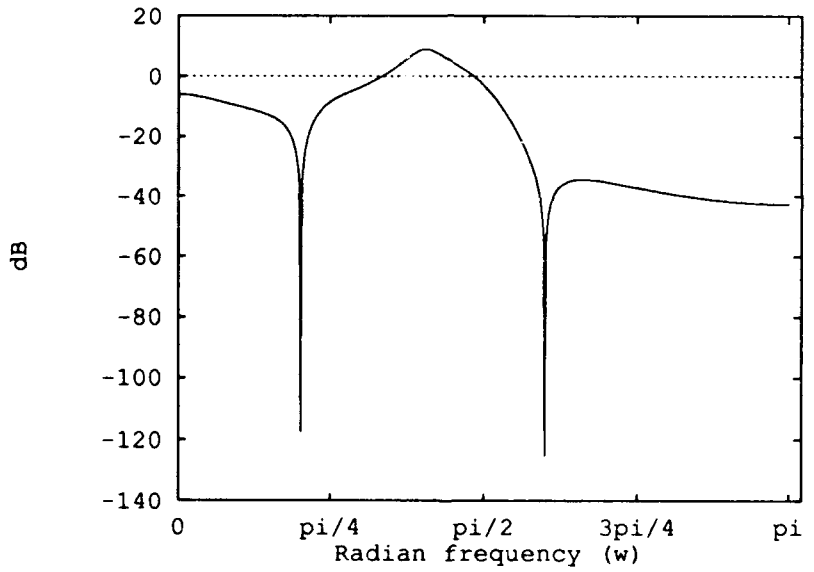


Figure 5.63. Magnitude Plot, 8-bits to represent coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I.

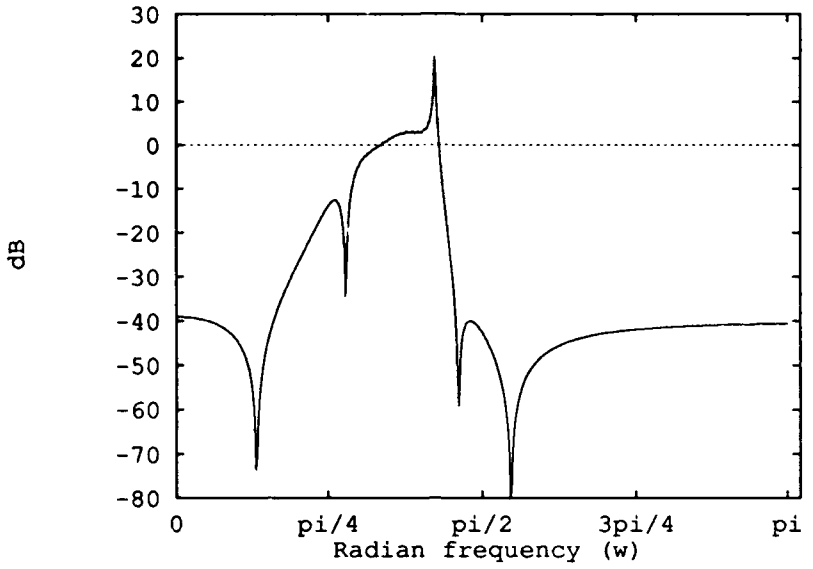


Figure 5.64. Magnitude Plot, 16-bits to represent coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I.

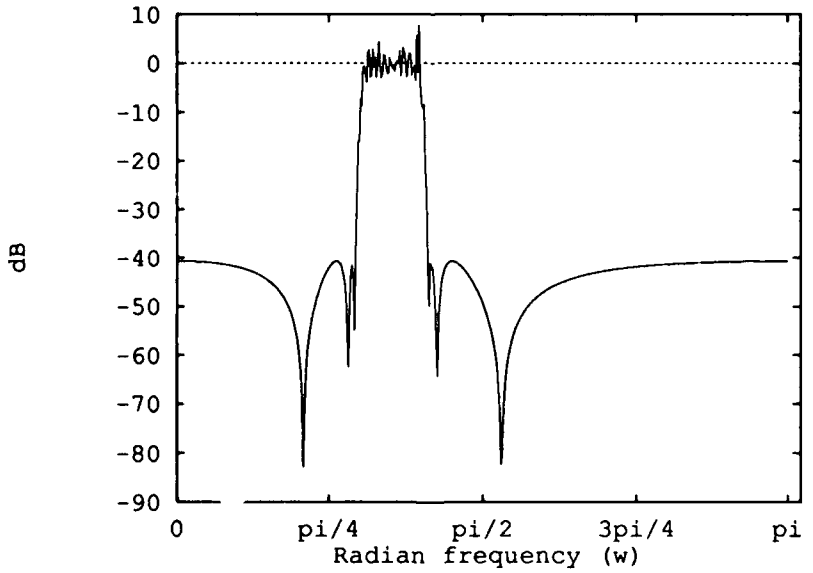


Figure 5.65. Magnitude Plot, 24-bit Single Precision coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I.

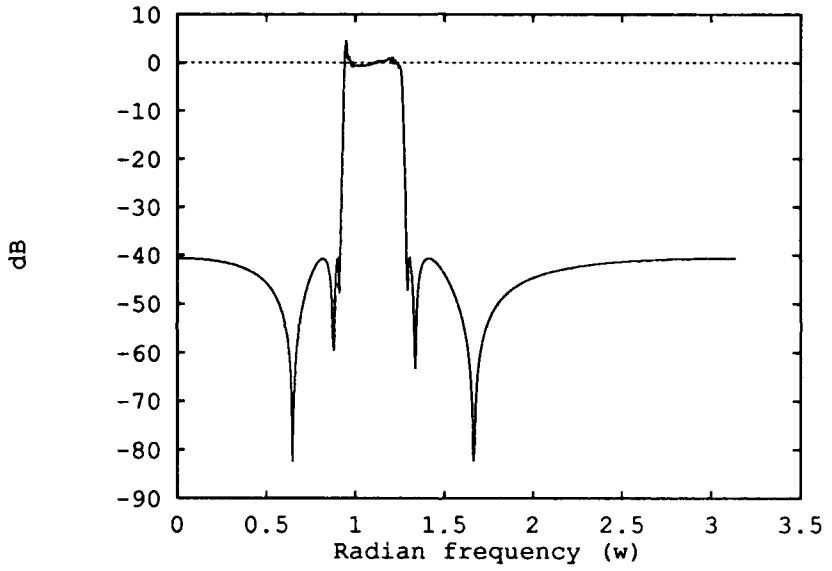


Figure 5.66. Magnitude Plot, 48-bit Double Precision for coefficients in Table 5.12, normalized conversion from the cascade structure, direct form I.

5.8 Summary

The results of the digital filter simulator tool was presented. Examples of FIR, IIR, digital filters implemented in both the direct and second order cascade structures were shown. Output plots in magnitude, phase, and error were given using various resolutions to represent the coefficients. It was shown that the use of the second order cascade filter structure is more resilent to the effects of coefficient quantization when the order of the filter is above five. Each filter design needs to be simulated in order to find the best performance. It was found that sometimes less bits could be used to improve the transfer characteristics. Such odd results can be found by the use of the simulator. Linear phase characteristics of filters were maintained with coefficient quantization. Simulation can bring out the execptions and the problems in realizations.

VI. Conclusions and Recommendations For Further Study

This chapter summarizes and discusses the results obtained from the digital filter simulation tool designed for this thesis. The coefficient quantization effects on the frequency response of a digital filter have been described. This was accomplished by changing the coefficient values for implementation using the restraint of (B+1) bits to represent the coefficients. The roundoff noise generated within a digital filter was found by using the impulse response. Simulating the effects of roundoff provided a means to compute roundoff power. The direct and second-order cascade structures were used for the simulation of finite precision effects. The software tool is user friendly, menu driven, and provides help screens with minimal keystrokes required by a user. Results of Chapter 5 are presented and compared to other works. Conclusions are made to formulate a strategy in the implementation of digital filters.

6.1 Conclusions

The digital analyzer software tool provides the designer with a means to exemplify the restraining conditions imposed by the quantization of the coefficients and subsequent roundoff errors generated by the realization. The designer will constantly desire to reduce the complexity and cost of the digital filter. This software tool provides a means to experimentally determine the number of bits for number representations to meet design specifications.

6.1.1 Coefficient Representation The examples of Chapter 5 show the effects on a system transfer function when finite-precision effects are accounted for. The examples show that the use of 16-bits to represent the coefficients will generally provide the desired system response. The system transfer response can change dramatically when 8-bits or less is used. This simulation tool allows a designer to see and measure the transfer function when implemented with finite-precision effects. Anomilies of the system transfer function can also occur. With some filter implementations, using cascade over direct will not improve performance. Other filter examples show the use of fewer bits can improve the pass band characteristics.

6.1.2 Analysis of Results The order of the difference equation can affect the transfer function. The use of high-order equations (generally those greater than second-order) to specify a frequency response for a digital filter are implemented in second-order sections.

When the difference equation in z-domain is solved using higher-order powers, the amount of the error (quantization) from the unquantized number to the restricted quantized position in the z-plane is also raised to the higher order. This will tend to further degrade the performance. This effect is shown by comparing the system transfer function from different implementation structures.

The placement of the poles and zeros in the z-plane also has a direct relation to the amount of quantization error produced. When poles or zeros with some non-zero magnitude are located close to the real axis in the z-plane, the amount of quantization will be greater than if the position of the poles or zeros is further away from the real axis. The number of allowed quantized positions in the z-plane is less dense when close to the real axis. Likewise, the number of allowed positions is more dense away from the real axis with increasing magnitude.

In FIR filter designs, the zeros are more uniformly placed in the z-plane than in IIR filter designs [15:345-351]. If the zeros are tightly clustered in one area in the z-plane, then the sensitivity will increase from the effects of coefficient quantization. Small movements in the zeros will cause larger changes in the relative distance between them than if the zeros were uniformly distributed. FIR designs then, due to the uniformity of zeros in the z-plane, will have less sensitivity to coefficient quantization than other designs with clustered poles and zeros.

Choosing the cascade structure over the use of the direct structure is a best choice when using filters with more than three or four delays. The benefits are two fold. First, the structure is simpler. The number of computations needed to compute an IIR filter with two delays in the feed forward direction and two delays in the feed back direction are five multiplies, one add, and four delays. A comparable cascade implemented filter will have five multiplies, two adds, and only two delays. When the order of the difference equation for the transfer function increases, the difference in the number of delay elements will increase as well. Second, the amount of degradation will be less. The use of only second-order sections will withstand quantization errors better than higher order polynomials as shown in Section 5.7.

6.2 Recommendations For Further Study

Any time a software tool is developed, the areas for further enhancement will include refinements and additional capability. This case is no different. The digital filter software analyzer could have some enhancements made to it. Logical extensions for future work are:

1. Investigate the estimation of roundoff power using other forms of input besides an impulse. Changing the length of the impulse response to attain a minimum value instead of taking 100 output samples can be done.
2. Enhance the use of information hints when an error occurs with input and output. Currently, the software will move around problems such as a file doesn't exist and re-display the main menu or just give a simple 'be careful' message to the user.
3. Remove the constraint for equal number of cascade sections in the numerator and denominator of the transfer function. This constraint imposes that the user put in a few coefficients with zero magnitude.
4. Use graphics to show the pole/zero placement when implementing cascade structures to provide insight to the filter design.
5. Develop an interactive software tool with graphics to move the poles and zeros about in the z-plane and see the resulting transfer function. The conversion of this software to Quick Basic could accomplish these tasks and run on a PC/AT.
6. Add a complete smart filter designer to compute coefficients for a filter. This would allow the user to specify the tvpe of frequency response needed, the tool would generate the coefficients and then show the resulting transfer function.
7. Implement an adaptive filter to forward model a desired frequency response. Then a smart design tool could also generate filter coefficients. The questions to ask would be how well can the adaptive filter perform and what types of adjustments improve performance?

Appendix A. *Program Listing for Digital Software Analyzer Tool*

```
PROGRAM PC7
C*******************************************************************************
C--- FILE PC7
C--- WRITTEN BY CAPT PERRY L. CHOATE, SUMMER 1991
C---
C--- WORD REPRESENTATION IS TWO'S COMPLEMENT.
C--- WORD LENGTH IS (B + 1) BITS WHERE B IS BITS FOR NUMBER, PLUS SIGN BIT.
C---
C--- THIS PROGRAM ACCEPTS COEFFICIENTS OF A LINEAR CONSTANT COEFFICIENT DIFF
C--- ERENCE EQUATION AND CALCULATES MAGNITUDE, PHASE, AND GROUP DELAY RE-
C--- SPONSES.
C---
C--- *****   *****   ***** DIRECT FORM   *****   *****   *****
C--- THE FORM FOR ENTERING THE DIFFERENCE EQUATION IS THAT THE COEFFICIENTS
C--- FOR THE OUTPUT SAMPLES ARE ON ONE SIDE OF THE EQUAL SIGN AND THE
C--- COEFFICIENTS FOR THE INPUT SAMPLES ARE ON THE OTHER SIDE OF THE
C--- EQUAL SIGN.  THIS SHOULD RESOLVE THE SIGNS OF THE COEFFICIENTS.
C---
C THE X COEFFICIENTS ARE ON THE RIGHT SIDE OF DIFFERENCE EQUATION'
C THE Y COEFFICIENTS ARE ON THE LEFT SIDE OF DIFFERENCE EQUATION'
C FROM OPPENHEIM AND SCHAFER.................
C
C          ************** EXAMPLE ***************
C             EACH FILTER LOOKS LIKE THIS
C
C     Y(Z)*A_0 + Y(Z^-1)*A_1 + Y(Z^-2)*A_2 + ...   =
C
C     X(Z)*B_0 + X(Z^-1)*B_1 + X(Z^-2)*B_2 + ...
C
C
C--- *****   *****   ***** CASCADE FORM   *****   *****   *****
C--- ENTER CASCADE SECTIONS WITH THREE COEFFICIENTS IN THE NUMERATOR AND
C--- THREE COEFFICIENTS IN THE DENOMINATOR.  MAKE SURE THE DENOMINATOR
C       TERM  A_0 * Z^0   HAS A_0=1.0!
C   BE SURE TO INPUT COEFFICIENTS LIKE IN EQ. 6.23
C    FROM OPPENHEIM AND SCHAFER.................
C   BUT INCLUDE THE '1.0' IN THE DENOMINATOR AS A COEFFICIENT
C   EACH SECTION WILL HAVE SIX COEFFICIENTS.................
C        ************** EXAMPLE ***************
C              EACH SECTION IS LOOKS LIKE THIS
```

```
C
C          (B_0 + B_1*Z^-1 + B_2*Z^-2)   (...)
C H(Z) =   ------------------------  * ----- * ...
C  (A_0 - A_1*Z^-1 - A_2*Z^-2)   (...)
C
C                                  note:  A_0 = 1.0 always
C
C---
C--- INPUT:  THE X COEFFICIENTS
C---         THE Y COEFFICIENTS
C---   NUMBER OF TERMS OF X(N-K)
C--- NUMBER OF TERMS OF Y(N-K)
C--- DIFFERENCE EQUATION FROM FILE (OPTIONAL)
C--- NUMBER OF BITS TO REPRESENT THE NUMBER
C--- FILE NUMBER TO ASSOCIATE WITH ALL OUTPOUT FILES
C---
C---
C--- OUTPUT: MAGNITUDE RESPONSE FILE 'MAG#.DAT'
C---   PHASE RESPONSE FILE 'PHASE#.DAT'
C--- GROUP DELAY RESPONSE FILE 'GPDELAY.DAT'(OUTPUT NOT GENERATED)
C--- OUTPUT: QUANTIZED MAGNITUDE RESPONSE FILE 'MAGQ#.DAT'
C---   QUANTIZED PHASE RESPONSE FILE 'PHASEQ#.DAT'
C--- GROUP DELAY RESPONSE FILE 'GPDELAYQ#.DAT'(OUTPUT NOT GENERATED)
C--- MEAN SQUARE ERROR FILE 'ERROR#.DAT'
C--- ROUND-OFF POWER IN FILTER (SCREEN OUTPUT ONLY)
C--- THE MOTIVATION FOR THE APPROACH TO CALCULATING THESE RESPONSES IS FROM
C--- EQUATIONS 5.18 AND 5.46, PAGES 206 AND 213, OF OPPENHEIM AND SCHAFER
C---
C HERE I DECLARE THE VARIABLES USED WITHIN THE MAIN PROGRAM.
C
REAL    ERROR(2048),XAXISTEP(2048),UNQUANTMAG(2048),QUANTMAG(2048)
REAL  PI,MAG_H(2048),PHA_H(2048),MAGQ_H(2048),PHAQ_H(2048)
REAL A(512),B(512),AQ(1028),BQ(1028),GPDELAY(2048),GPDELAYQ(2048)
REAL LOWER, STEP, RANGE, W
INTEGER NUMBITS,NPOINTS,FILTERTYPE,CONTROL
COMPLEX NUM,DEN,H
BYTE ANS,CFLAG

DOUBLE PRECISION DATA(2048)
CHARACTER*15 FNAME, FILENAME, FILTERNAME
CHARACTER*2 C2,FNUMBER, UNQFNAME
CHARACTER*12 C4
CHARACTER*3 C1
CHARACTER*4 C3,C7
CHARACTER*5 C5
```

```
CHARACTER*6 C6

C
C INITIALIZATION OF VARIABLES IS DONE UPON ENTRY INTO THE MAIN ROUTINE.
C

UNQFNAME = '01'
CONTROL = 0
FNUMBER = '99'
FNAME = '???????'
PI = 3.14159265358
STEP = 0.00628318
FILTERTYPE = 1
FILTERNAME = 'DIRECT FORM'
NPOINTS = 500
NUMBITS = 16
RANGE = PI
LOWER = 0.0
C

C*****************************************************************************
C-----------------------------------------------------------------------------
C THE DO WHILE IS A MEANS TO CONTINUE TO GET BACK TO THE MAIN MENU.
C THE MAIN MENU IS WHERE A USER WILL MAKE ALL THE ACTION HAPPEN.
C THE DO WHILE RUNS THE LENGTH OF THE MAIN PROGRAM.
C


DO WHILE ( CONTROL .GE. 0)


40 CONTINUE

WRITE(6,4)
WRITE(6,1) '*DISCRETE SIGNAL PROCESSING, FILTER DESIGN, Capt P. Choate*'
WRITE(6,4)

WRITE(6,5) '(1)  ENTER TYPE OF FILTER, OR CONVERSION PROCESS       :...'
WRITE(6,30) '     TYPE OF FILTER DESIRED IS ..... ',FILTERNAME

WRITE(6,5) '(2)  ENTER NUMBER OF BITS FOR COEFFICIENTS QUANTIZATION: ...'
WRITE(6,31) '     NUMBER BITS FOR COEFFICIENT QUANTIZATION IS.....',NUMBITS
WRITE(6,5) '(3)  ENTER NUMBER TO ASSOCIATE WITH THE OUTPUT FILES : ... '
WRITE(6,32)'     TWO DIGIT NUMBER TO TAG WITH OUTPUT FILES IS ..... ',FNUMBER
```

```
WRITE(6,5) '(4)  ENTER NUMBER FOR UNQUANTIZED MAG/PHASE OUTPUT FILES : ... '
WRITE(6,32) '     TWO DIGIT NUMBER FOR UNQUANTIEZED MAG/PHASE
     &                                                    FILES IS.. ',UNQFNAME

WRITE(6,5) '(5)  ENTER COEFFICIENTS TO USE FROM FILE OR KEYBOARD : ...'
WRITE(6,33) '     COEFFICIENTS' FILE NAME IS..... ',FNAME

WRITE(6,5) '(6)  ENTER NUMBER OF SPECTRAL POINTS FOR CALCULATIONS : ...'
WRITE(6,34) '     NUMBER OF SPECTRAL POINTS FOR CALCULATIONS IS: ...',NPOINTS

WRITE(6,5) '(7)  ENTER THE NYQUIST BANDWIDTH TO VIEW FOR PLOTTING : ...'
WRITE(6,35) '     THE BANDWIDTH TO VIEW FOR PLOTS IS.....',RANGE

WRITE(6,5)'(8)  CHECK COEFFICIENT FILE AND NORMALIZE IF NECESSARY: ...'
WRITE(6,36) '     WILL ALSO RENAME THE COEFFICIENT FILE.'

WRITE(6,5) '(9)  SAVE MY COEFFICIENTS TO A FILE IN THIS DIRECTORY :...'
WRITE(6,37) '     FOR KEYBOARD ENTERED COEFFICIENTS.'

WRITE(6,5) '(10)  FIND THE ROUND-OFF POWER ERROR FOR CASCADE .... :...'
WRITE(6,37) '     AND DIRECT DESIGNS; SWAP CASCADE SECTIONS  ....     '

WRITE(6,37)'(11)  HELP ON HOW TO FORMAT INPUT COEFFICIENT FILES. '

WRITE(6,5) '(55)   ------------   ALL DONE  ---------- '
WRITE(6,1)'*****   PLEASE ENTER NUMBER OR [RET] TO RUN PRGRAM   *****'
WRITE(6,1)'                                      ENTER NUMBER [#]:'

READ(5,38,END=99,ERR=40)CONTROL


C ************************************************************************
C THE CONTROL PICKS OUT THE SUB-ROUTINE TO RUN.  ONCE THE SUBROUTINE
C HAS RETURNED, THE USER CAN SELECT FROM THE MENU AGAIN.
C ************************************************************************

IF  ( CONTROL .EQ. 1 ) THEN
CALL TYPEOFFILTER(FILTERTYPE,FILTERNAME,FNAME,NX,NY,A,B)
ENDIF
IF  ( CONTROL .EQ. 2 ) THEN
CALL BITSAVAILABLE(NUMBITS)
ENDIF
IF  ( CONTROL .EQ. 3 ) THEN
CALL FILENUMBER(FNUMBER)
ENDIF
```

```
IF  ( CONTROL .EQ. 4 ) THEN
CALL UNQNUMFILE(UNQFNAME)
ENDIF

IF  ( CONTROL .EQ. 5 ) THEN
CALL READCOEFF(NX,NY,A,B,FNAME,ANS,FILTERTYPE)
CFLAG = 'Y'
ENDIF
IF  ( CONTROL .EQ. 6 ) THEN
CALL ITERATIONS(STEP,NPOINTS,RANGE)
ENDIF
IF  ( CONTROL .EQ. 7 ) THEN
CALL SETRANGE(STEP,NPOINTS,RANGE,LOWER)
ENDIF
IF  ( CONTROL .EQ. 8 ) THEN
CALL NORMALIZATION(FNAME,NX,NY,A,B)
ENDIF
IF  ( CONTROL .EQ. 9 ) THEN
CALL SAVECOEF(FNAME,NX,NY,A,B)
ENDIF
IF  ( CONTROL .EQ. 10 ) THEN
CALL ROUNDOFF(A,B,NX,NY,FILTERTYPE,NUMBITS)
ENDIF
IF  ( CONTROL .EQ. 11 ) THEN
CALL HELPONINPUT
END IF
IF  ( CONTROL .EQ. 55 ) THEN
GO TO 99
ENDIF

C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C THIS IF STATEMENT RUNS THE ENTIRE LENGTH OF THE MAIN ROUTINE.
C IF A USER HITS 'RETURN'  "CONTROL = 0" THEN THE PROGRAM WILL
C CALCULATE WITH DATA VALUES THE RESPONSE OF THE FILTERS
C (UN-QUANTIZED AND QUANTIZED).
C

IF  ( CONTROL .EQ. 0 ) THEN
C
C CFLAG  TELLS ME IF THE USER HAS INPUT A COEFFICIENT FILE OR NOT.
C        THE PROGRAM IS NOT RUN UNLESS A COEFFICENT FILE IS BUILT.
C
  IF ( CFLAG .EQ. 'Y' ) THEN
CONTINUE
```

```
ELSE
WRITE(6,1)'USER OF THE PROGRAM, INPUT COEFFICIENTS!'
GO TO 40

END IF



c---    I use up to pi radians to avoid the repitition in output.
C--- THE OUTPUT IS DESIGNED TO SHOW THE NYQUIST DYNAMIC RANGE ONLY.
C--- HENCE, THE MOST ACCURATE PICTURES OF THE PASS BAND CAN BE SHOWN.
C---
C--- THE VALUES OF THE NUMERATOR AND DENOMINATOR ARE CALCULATED
C--- THESE ARE COMPLEX VALUES.


C ***************************************************************************
C ***************************************************************************
C--- LOWER IS THE STARTING POINT FOR THE RANGE OF RADIAN VALUES.
C--- W IS THE POINT IN RADIAN FREQUENCY TO EVALUATE.
C--- STEP IS AMOUNT OF RADIAN FREQUENCY CHANGE (FOR NEXT CYCLE).
C--- XAXISTEP(*) IS ARRAY FOR RADIAN FREQUENCY USED FOR OUTPUTING.

C---------------------------------------------------------------------------
C THIS BEGINS THE LOOPING TO EVALUATE THE FREQUENCY POINTS
C FOR THE FILTER.  THE PROCESS EVALUATES ONE POINT AT A TIME AND
C THEN FROM THE ARRARY'S BUILT, THE PROGRAM WILL STORE THE RESULTS.
C
C PROCESS ONE
C
C
C

C###########################################################################
77 DO J = 1,NPOINTS

W = STEP * (J-1) + LOWER

C--- THE XAXISTEP(J) IS TO PLOT ON GNUPLOT WITH THE X-AXIS IN RADIANS.

XAXISTEP(J) = W

C ***************************************************************************
C THIS SECTION PERFORMS CALCULATIONS ON THE UN-QUANTIZED COEFFIECIENTS.
```

```
C A FILTER CAN BE EITHER IN DIRECT FORM OR IT CAN BE IN CASCADE FORM.
C FILTERTYPE DETERMINES THE FILTER STRUCTURE.


 IF ( FILTERTYPE .EQ. 1 ) THEN

CALL DIRECTDEN(W,NY,A,DEN)
CALL DIRECTNUM = DIRECTNUM(W,NX,B,NUM)

ELSE
CALL CASCADEDEN(W,NY,A,DEN,NUMBITS)
CALL CASCADENUM(W,NX,B,NUM,NUMBITS)

ENDIF

C ***************************************************************************


C--- CALCULATE THE MAGNITUDE RESPONSE

 IF ( (ABS(DEN)) .EQ. (0.0) ) THEN
WRITE(6,1)'THE DENOMINATOR IS ZERO!!!!!!!!!!!!!!!!!!'
ENDIF

C ***************************************************************************


    IF ( DEN .NE. (0.0,0.0)  ) THEN

      H = NUM/DEN
      MAG_H(J) = ABS(H)

   ELSE

MAG_H(J) = 1.1
        PHA_H(J) = PI/2
   ENDIF
C ***************************************************************************



C--- INTERMEDIATE STORAGE IS NEEDED TO CALCULATE THE LINEAR ERROR.
C--- THE MAGNITUDE IS FOUND IN DB NEXT, BUT LINEAR ERROR IS NEEDED.

UNQUANTMAG(J) = MAG_H(J)
```

```
C--- CALCULATE PHASE WITH ARC TANGENT FUNCTION
C--- THIS FUNCTION WAS TESTED AND FOUND TO YEILD THE CORRECT RESULTS
C--- FOR THE INPUTS GIVEN IN ALL QUADRANTS.
C---
C--- THE REAL AND IMAGINARY PARTS ARE EXTRACTED FROM THE COMPLEX NUMBER.
C---
C--- PHASE ANGLE CALCULATION FOR THIS STEPPED FREQUENCY.


   Y = AIMAG(H)
   X = REAL(H)

   PHA_H(J) =  ATAN2(Y,X)


END DO
C*************************************************************************

C-------------------------------------------------------------------------
C ALL THE ARRAYS ARE BUILT FOR STORAGE AS A FILE.
C THE UNQUANTIZED VALUES ARE USED TO COMPARE TO THE QUANTIZED
C COMPUTATIONS DONE NEXT.
C ************************************************************************
C ************************************************************************

C--- NOW USE THE RESULTS OF PHASE RESPONSE TO CALCULATE THE GROUP DELAY
C--- APPROXIMATE THE DERIVATIVE WITH A DIFFERENCE
C---    THE GROUP DELAY IS FOUND BUT NOT SAVED AT THIS POINT.

DO J = 1,NPOINTS-1
   GPDELAY(J)=-1.0*(PHA_H(J+1)-PHA_H(J))/(2.0*PI/NPOINTS)
END DO


C ************************************************************************
C--- WRITE OUTPUT DATA FILE FOR MAGNITUDE RESPONSE
C--- THE NAME OF THIS FILE WILL BE MAG.DAT.  SINCE THIS IS THE
C--- UNQUANTIZED MAGNITUDE PLOT.  THE NUMBER WILL HELP
C1 = 'mag'
C2 = UNQFNAME
C3 = '.dat'
C4 = C1//C2//C3
FILENAME = C4
```

```
OPEN(13,NAME=FILENAME,STATUS='UNKNOWN',ERR=40)
DO J = 1,NPOINTS

C--- BOUNDS ARE SET SO GNUPLOT WILL NOT BE OUT OF RANGE FOR PLOTTING
  IF ( MAG_H(J) .LT. (1.0E-10) ) THEN
MAG_H(J) = (1.0E-10)
ENDIF

C--- THE MAGNITUDE IS PUT INTO LOGRITHMIC FORM.
  IF ( MAG_H(J) .EQ. (0.0) )THEN
MAG_H(J) = -1*12.*10.
ELSE
MAG_H(J) = 20.0 * ALOG10(MAG_H(J))
ENDIF
           write(13,15,ERR=40)XAXISTEP(J),MAG_H(J)


END DO


CLOSE(13)
C ***************************************************************************


C ***************************************************************************
C--- PUT PHASE INFO TO FILE FOR PLOTS

C5 = 'phase'
C2 = UNQFNAME
C3 = '.dat'
C4 = C5//C2//C3
FILENAME = C4


OPEN(14,NAME=FILENAME,STATUS='UNKNOWN',ERR=40)

DO J = 1,NPOINTS

C--- BOUNDS ARE SET SO GNUPLOT WILL NOT BE OUT OF RANGE FOR PLOTTING
  IF ( ABS(PHA_H(J)) .LT. (1.0E-10) ) THEN
PHA_H(J) = 0.0
ENDIF


   DATA(J) = PHA_H(J)
```

```
   write(14,15,ERR=40)XAXISTEP(J),PHA_H(J)


END DO
CLOSE(14)
C ***************************************************************************

C ***************************************************************************
C ******************      QUANTIZATION STARTS       **********************
C ***************************************************************************

C ***************************************************************************
C---    THE QUANTIZATION IS APPLIED STARTING HERE ---
C--- THIS SECTION CALLS A FUNCTION CALLED QUANT.
C--- QUANT TRUNCATES THE NUMBER GIVEN TO A FORM EQUAL TO THE NUMBER OF BIT
C--- REPRESENTATION AVAILABLE.  QUANT THEN RETURNS THE NEW QUANTIZED NUMBER.
C---
C ***************************************************************************
C
C THE COEFFICIENTS ARE QUANTIZED HERE!!!!
C
DO J = 1, NY
AQ(J) = QUANT(A(J),NUMBITS)
END DO



DO J = 1, NX
BQ(J) = QUANT(B(J),NUMBITS)
END DO

C-------------------------------------------------------------------------
C THIS STARTS THE LOOPING FOR EVALUATION EACH POINT ALONG THE
C FREQUENCY AXIS.  THIS WILL EVALUATE THE NUMBER OF POINTS THAT
C THE USER REQUESTED AND THE RANGE OF VALUES THAT THE USER
C INPUT TO THE PROGRAM.

C#########################################################################

87 DO J = 1,NPOINTS

W = STEP * (J-1) + LOWER

C--- THE XAXISTEP(J) IS TO PLOT ON GNUPLOT WITH THE X-AXIS IN RADIANS.
```

```
XAXISTEP(J) = W


C ***************************************************************************
C---  THE QFLAG SIGNALS THE CASCADE ROUTINES TO PERFORM QUANTIZATION OR NOT
C--- TO PERFORM QUANTIZATION.  THE QUANTIZATION IS APPLIED AFTER EACH SECTION
C--- OF THE CASCADE FILTER IS CALCULATED BEFORE THE RESULT ENTERS THE NEXT
C--- SECTION. QFLAG = 'N' NO,   QFLAG = 'Y' FOR YES.


 IF ( FILTERTYPE .EQ. 1 ) THEN

CALL DIRECTDEN(W,NY,AQ,DEN)
CALL DIRECTNUM = DIRECTNUM(W,NX,BQ,NUM)

ELSE
C--- I DO CALL THE QUANTIZER ROUTINE IN THIS SECTION.
C--- RESULTS FROM EACH SECTION ARE QUANTIZED BEFORE ENTERING THE
C--- NEXT SECTION IN THE DIGITAL FILTER.

CALL CASCADEDEN(W,NY,AQ,DEN,NUMBITS)
CALL CASCADENUM(W,NX,BQ,NUM,NUMBITS)

ENDIF



C--- CALCULATE THE MAGNITUDE RESPONSE
C ***************************************************************************


 IF ( (ABS(DEN)) .EQ. (0.0) ) THEN
WRITE(6,1)'THE DENOMINATOR IS ZERO IN QUANTIZE PART!!!!'
ENDIF


C ***************************************************************************


    IF ( DEN .NE. (0.0,0.0)  ) THEN

      H = NUM/DEN
      MAGQ_H(J) = ABS(H)

   ELSE
```

```
MAGQ_H(J) = 1.1
        PHAQ_H(J) = PI/2
   ENDIF


C ***************************************************************************


C--- INTERMEDIATE STORAGE IS NEEDED TO CALCULATE THE LINEAR ERROR.
C--- THE MAGNITUDE IS FOUND IN DB NEXT, BUT LINEAR ERROR IS NEEDED.

QUANTMAG(J) = MAGQ_H(J)



C--- CALCULATE PHASE WITH ARC TANGENT FUNCTION
C--- THIS FUNCTION WAS TESTED AND FOUND TO YEILD THE CORRECT RESULTS
C--- FOR THE INPUTS GIVEN IN ALL QUADRANTS.
C---
C--- THE REAL AND IMAGINARY PARTS ARE EXTRACTED FROM THE COMPLEX NUMBER.
C---
C--- PHASE ANGLE CALCULATION FOR THIS STEPPED FREQUENCY.

   Y = AIMAG(H)
   X = REAL(H)

   PHAQ_H(J) =  ATAN2(Y,X)

END DO

C###########################################################################

C---------------------------------------------------------------------------
C THIS ENDS THE LOOPING FOR EVALUATION EACH POINT ALONG THE
C FREQUENCY AXIS.
C
C ***************************************************************************
C ***************************************************************************


C ***************************************************************************
C--- NOW USE THE RESULTS OF PHASE RESPONSE TO CALCULATE THE GROUP DELAY
C--- APPROXIMATE THE DERIVATIVE WITH A DIFFERENCE
C---    THE GROUP DELAY IS FOUND BUT NOT SAVED AT THIS POINT.
C ***************************************************************************
```

```
DO J = 1,NPOINTS-1
   GPDELAYQ(J)=-1.0*(PHAQ_H(J+1)-PHAQ_H(J))/(2.0*PI/NPOINTS)
END DO

C ***************************************************************************
C--- WRITE OUTPUT DATA FILE FOR MAGNITUDE RESPONSE

C7 = 'magq'
C2 = FNUMBER
C3 = '.dat'
C4 = C7//C2//C3
FILENAME = C4


OPEN(15,NAME=FILENAME,STATUS='UNKNOWN',ERR=40)

DO J = 1,NPOINTS

C--- BOUNDS ARE SET SO GNUPLOT WILL NOT BE OUT OF RANGE FOR PLOTTING
C--- THE MAGNITUDE IS PUT INTO LOGRITHMIC FORM.
 IF (  MAGQ_H(J) .LT. (1.0E-10)  ) THEN
MAGQ_H(J) = (1.0E-10)
ENDIF

 IF (  MAGQ_H(J) .EQ. (0.0)  ) THEN
MAGQ_H(J) = -1*12.*10.
ELSE
MAGQ_H(J) = 20.0 * ALOG10(MAGQ_H(J))
ENDIF

           write(15,15,ERR=99)XAXISTEP(J),MAGQ_H(J)


END DO

close(15)

C ***************************************************************************


C ***************************************************************************
C--- PUT PHASE INFO TO FILE FOR PLOTS

C6 = 'phaseq'
```

```
C2 = FNUMBER
C3 = '.dat'
C4 = C6//C2//C3
FILENAME = C4


OPEN(16,NAME=FILENAME,STATUS='UNKNOWN',ERR=40)

DO J = 1,NPOINTS

C--- BOUNDS ARE SET SO GNUPLOT WILL NOT BE OUT OF RANGE FOR PLOTTING
 IF (  ABS(PHAQ_H(J)) .LT. (1.0E-10) ) THEN
PHAQ_H(J) = 0.0
ENDIF


   write(16,15,ERR=40)XAXISTEP(J),PHAQ_H(J)

END DO

CLOSE(16)

C *************************************************************************

C--- ERROR IS FOUND AS A LINEAR ERROR.  RESULTS FROM THE MAGNITUDE SECTION
C---    ARE USED FOR THIS CALCULATION.
C---  THE ERROR IS FOUND BETWEEN THE UNQUANTIZED VERSION AND THE QUANTIZED.

DO J = 1,NPOINTS
ERROR(J) = (UNQUANTMAG(J) - QUANTMAG(J))
END DO


C *************************************************************************
C--- WRITE OUTPUT DATA FILE FOR LINEAR ERROR

C5 = 'error'
C2 = FNUMBER
C3 = '.dat'
C4 = C5//C2//C3
FILENAME = C4



OPEN(17,NAME=FILENAME,STATUS='UNKNOWN',ERR=40)
```

```
DO J = 1,NPOINTS
  IF ( ABS(ERROR(J)) .LT. (1.0E-10) ) THEN
ERROR(J) = 0.0
ENDIF


          write(17,15,ERR=40)XAXISTEP(J),ERROR(J)

END DO

CLOSE(17)

C ***************************************************************************


C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C THE "ELSE  GOTO 40  ENDIF"  STATEMENTS ENDS THE MAIN ROUTINE.
C IF A USER HITS 'RETURN'  "CONTROL = 0" THEN THE PROGRAM WILL
C CALCULATE WITH DATA VALUES THE RESPONSE OF THE FILTERS
C (UN-QUANTIZED AND QUANTIZED). OTHERWISE THE USER CAN RUN ANY
C OF THE SUBROUTINES LISTED IN THE MENU.
C

ELSE

GOTO 40

ENDIF

C
C THE DO ENDS THE DO WHILE.  ALL THIS DOES IS CONTINUE TO CYCLE BACK
C TO THE MAIN MENU SO THE USER CAN MAKE FURTHER CHOICES FOR ACTION.

END DO




C ***************************************************************************
C ***************************************************************************

1 FORMAT(/,A,$)
2 FORMAT(I)
```

```
3 FORMAT(E17.10)
4 FORMAT(/)
5 FORMAT(A)
7 FORMAT(I,2X,I)
8 FORMAT(I,2X,G,2X,G)
9  FORMAT(I2,2X,E15.10,2X,E15.10)
10 FORMAT(2X,'A(',I1,')= ',$)
11 FORMAT(2X,'B(',I1,')= ',$)
12  FORMAT('The number of BITS USED is =',I3)
15 FORMAT(F5.3,4X,E17.9)
16 FORMAT('THE VALUE NPOINTS IS =',I5)
17 FORMAT(/,'THE NAME OF THE COEFFICIENT FILE IS CALLED "',A,'"')
20 FORMAT(2X,'A(',I3,') = ',F21.10)
21 FORMAT(2X,'B(',I3,') = ',F21.10)
22 FORMAT(2X,'A(',I3,') = ',G)
23 FORMAT(2X,'B(',I3,') = ',G)
24 FORMAT('IMG=',F15.10,2X,'REAL=',F15.10,2X,'ATAN=',E17.9)
25 FORMAT('BITS=',I3,2X,'QUANT=',E17.10,2X,'VALUE=',E17.10)
26 FORMAT('BITS=',I3,2X,'QUANT=',G,2X,'VALUE=',G)
27 FORMAT(2X,/,'PHASE IS = (',G,')',/)
28 FORMAT('IMAGINARY =',F12.8,1X,'REAL =',F12.8,1X,'ATAN =',F14.9)
29 FORMAT(/,'VALUE IS',F20.10)
30 FORMAT(A,A,/)
31  FORMAT(A,I3,/)
32 FORMAT(A,A,/)
33 FORMAT(A,A,/)
34 FORMAT(A,I4,/)
35 FORMAT(A,F7.4,1X,'RAD/SEC',/)
36 FORMAT(A,/)
37 FORMAT(A,/)
38 FORMAT(I,$)
39  FORMAT(I3,2X,E12.5,2X,E12.5,2x,F5.2)

99 STOP
END


C ***************************************************************************


C ***************************************************************************
C THIS FUNCTION IS USED TO FIND THE DECIMAL VALUE TO THE ACCURACY OF
C THE NUMBER OF BITS AVAILBLE.
C QVALUE IS THE INPUT NUMBER IN 23-BITS ACCURACY.
C QUANT IS THE VALUE RETURNED - A NUMBER WITH NUMBITS ACCURACY.
```

```
C
C
FUNCTION QUANT(QVALUE,NUMBITS)
REAL MAXVALUE, MINVALUE,HIGHNUM,QVALUE,VALUE
INTEGER TRUNCATED


MAXVALUE = 1.0
MINVALUE = -1.0

 IF ( QVALUE .GT. MAXVALUE ) THEN
QUANT = MAXVALUE
WRITE(6,4)
WRITE(6,1) 'WARNING::::::USE NORMALIZATION ROUTINE.....'
WRITE(6,5)QUANT,MAXVALUE
WRITE(6,4)

END IF

IF ( QVALUE .LT. MINVALUE ) THEN
QUANT = MINVALUE
WRITE(6,4)
WRITE(6,1) 'WARNING::::::USE NORMALIZATION ROUTINE.....'
WRITE(6,7)QVALUE,MINVALUE
WRITE(6,4)

END IF

IF ( QVALUE .GE. 0.0 ) THEN

HIGHNUM = (2** (NUMBITS - 1) )
VALUE = ((HIGHNUM*QVALUE)+0.5)
TRUNCATED = IFIX(VALUE)
QUANT = (TRUNCATED/HIGHNUM)
IF ( QUANT .EQ. (0.0) ) THEN
WRITE(6,8)
END IF

ELSE

HIGHNUM = (2** (NUMBITS - 1) )
VALUE = ((HIGHNUM*QVALUE)-0.5)
TRUNCATED = IFIX(VALUE)
QUANT = (TRUNCATED/HIGHNUM)
IF ( QUANT .EQ. (0.0) ) THEN
```

```
WRITE(6,8)
END IF


ENDIF


c WRITE(6,1) 'The function:'
c write(6,20)numbits,QUANT,QVALUE
1 FORMAT(A,$)
4 FORMAT(/)
5 FORMAT(/,'THE VALUE TO QUANTIZED IS =',E12.6,'   THE MAXVALUE =',F7.3)
7 FORMAT(/,'THE VALUE TO QUANTIZED IS =',E12.6,'   THE MINVALUE =',F7.3)
8 FORMAT(/,'COEFFICIENT WAS JUST QUANTIZED TO A VALUE OF ZERO.')


RETURN
END

C ***************************************************************************
C ***************************************************************************

C ***************************************************************************
C THIS FUNCTION OPERATES VERY CLOSELY TO QUANT.  BUT IT IS CALLED BY
C TWO OTHER SUBROUTINES (ROUNDCASCADE AND ROUNDDIRECT).
C THESE SUBROUTINES DON'T REQUIRE THE CHECKS FOR NORMALIZATION,
C THE CHECKS FOR OUT OF RANGE VALUES, OR OTHER I/O HANDLING.
C THIS FUNCTION JUST RETURNS THE DECIMAL QUANTIZED VALUE OF THE NUMBER
C GIVEN TO IT.
C
FUNCTION ROUNDQ(QVALUE,NUMBITS)

REAL HIGHNUM,QVALUE,VALUE
INTEGER TRUNCATED

 IF ( QVALUE .GT. MAXVALUE ) THEN
QUANT = MAXVALUE
END IF

IF ( QVALUE .LT. MINVALUE ) THEN
QUANT = MINVALUE
END IF

IF ( QVALUE .GE. 0.0 ) THEN
```

```
HIGHNUM = (2** (NUMBITS - 1) )
VALUE = ((HIGHNUM*QVALUE)+0.5)
TRUNCATED = IFIX(VALUE)
ROUNDQ = (TRUNCATED/HIGHNUM)

ELSE

HIGHNUM = (2** (NUMBITS - 1) )
VALUE = ((HIGHNUM*QVALUE)-0.5)
TRUNCATED = IFIX(VALUE)
ROUNDQ = (TRUNCATED/HIGHNUM)

ENDIF

RETURN
END

C ***************************************************************************
C ***************************************************************************
C
C THIS SUBROUTINE IS CALLED BY THE MAIN MENU.
C ROUNDOFF IS HANDLED BY THESE NEXT THREE ROUINES.  IT'S A LITTLE BIT
C COMPLICATED, BUT NOT TOO BAD IF YOU TAKE THINGS ONE STEP AT A TIME.
C THE PURPOSE IS TO FIND THE ROUND OFF NOISE ERROR TERM FOR A CASCADE
C DESIGNED FILTER.  THIS ROUTINE FINDS THE IMPULSE RESPONSE FOR THE
C FILTER.  THEN THIS ROUTINE FINDS THE IMPULSE RESPONSE WHILE ACCONTING
C FOR ROUND-OFF ERROR.  THIS INFORMATION IS USED TO FIND THE NOISE
C POWER IN THE CASCADE FILTER AS IMPLEMENTED.
C ALSO, THE ROUND OFF ERROR CAN BE FOUND FOR A DIRECT FORM.  BUT THE
C DIRECT FORM ROUND OFF ERROR IS SIMPLER SINCE THE FORM OF IMPLEMENTATION
C IS SET OR NON-CHANGEABLE.
C
C THIS SUBROUTINE TAKES THE COEFFICIENTS, THE FILTER TYPE (CASCADE OR
C DIRECT) AND THE NUMBER OF BITS TO FIND THE ROUNDOFF ERROR.

SUBROUTINE ROUNDOFF(A,B,NX,NY,FILTERTYPE,NUMBITS)

REAL A(512), B(512)
BYTE ANS
INTEGER NX, NY, FILTERTYPE

C ---------------------------------------------------------------------
C CHECKS FOR DIRECT DESIGN.

IF (FILTERTYPE .EQ. 1) THEN
```

```
CALL ROUNDDIRECT(A,B,NX,NY,NUMBITS)
RETURN

END IF


C ---------------------------------------------------------------
C CHECKS FOR CASCADE DESIGN.

IF (FILTERTYPE .EQ. 2) THEN

CALL ROUNDCASCADE(A,B,NX,NY,NUMBITS)

END IF
C
C A USER MAY CHOOSE TO TRY A DIFFERENT ORDER IN THE CASCADE DESIGN.
C THE LOWEST ROUNDOFF POWER WILL OCCUR WHEN THE SECTION WITH THE HIGHEST
C GAIN IS PUT FIRST.
C

WRITE(6,1)'-------------------------------------------------------'
WRITE(6,1)'.......CHANGE THE CASCADE ORDER FOR COMPUTATIONS?......'
WRITE(6,1)'.......(Y)ES CHANGE THE CASCADE ORDER.................'
WRITE(6,1)'.......[ANY OTHER KEY] TO CONTINUE.....................    :'
READ(5,5)ANS

IF ( (ANS .EQ. 'y') .OR. (ANS .EQ. 'Y') )THEN
CALL CHANGECASCADEORDER(A,B,NX,NY,NUMBITS)
END IF
C ---------------------------------------------------------------
1 FORMAT(/,A,$)
5 FORMAT(A,$)
98 RETURN
99 END
C--- END OF ROUND-OFF ROUTINE.


C ***************************************************************************
C ***************************************************************************
C THIS SUBROUTINE IS CALLED BY ROUNDOFF.
C THIS SUBROUTINE FINDS THE OUTPUT FOR THE CASCADE STRUCTURE IN BOTH
C UNQUANTIZED AND QUANTIZED STRUCTURES.  THE DIFFERENCE BETWEEN THE TWO
C IS USED TO FIND THE POWER IN THE ROUNDOFF NOISE.
C
```

```
C NOTE: THIS ROUTINE USED THE CASCADE STRUCTURE (2D FORM), IN CASE
C YOU WANT TO FIGURE OUT HOW THIS WORKS.  THE INTERMEDIATE POINTS IN
C THE STRUCTURE ARE CALLED P(), P1(),P2().  THE BEST WAY TO SEE THIS IS
C TO LOOK IN THE THESIS AND SEE THE DIRECT FORM II STRUCTURE.
C I HAVE A SECTION ON HOW THE COMPUTATION IS DONE.  EACH NODE MUST BE
C CALCULATED AND THEN THE OUTPUT IS FOUND.  CYCLING TO THE NEXT STATE IS
C TRICKY SINCE NODES MUST COMPUATIONALLY RIPPLE UPWARDS SO AS NOT TO
C DESTROY THE NEXT STATE'S VALUE.
C
C THE P*(*) ARRAYS ALL CORRESPOND TO THE NODES IN THE STRUCTURE.
C THE PQ*(*) ARRAYS CORRESPOND TO THE QUANTIZED CALCULATED CASCADE FILTER.
C NY AND NX ARE THE NUMBER OF COEFFICIENTS.
C  DISPLAY ROUTINES MAKE THE SCREEN OUTPUT NICE TO READ.
C

SUBROUTINE ROUNDCASCADE(A,B,NX,NY,NUMBITS)

REAL P(100), P1(101), P2(101), Y(-2:100, 12), X(-2:100)
REAL A(512), B(512), MEAN(0:100), MEANSE
REAL PQ(100), P1Q(100), P2Q(100), YQ(-2:100, 12), ERROR(100), MEAN(100)
REAL SUMOFERRORSQ, NOISEPOWER
INTEGER NX, NY, NUMBITS, SECTIONS, POS1, POS2, POS3
INTEGER POINTS

POINTS = 100
C IF THIS SECTION IS ENTERIED A SECOND TIME, THIS WILL ZERO OUT ARRAYS
C THAT ARE USED WHEN VALUES ARE LEFT IN THEM.
DO I = 1, POINTS
X(I)  = 0.0
P(I)  = 0.0
P1(I) = 0.0
P2(I) = 0.0
END DO

X(1) = 1.0
MEAN(1) = 0.0
SUMOFERRORSQ = 0.0
P2(101) = 0.0
P1(101) = 0.0
SECTIONS = NX/3

JJ = 1
POS1 = ( (JJ * 3) - 2 )
POS2 = ( (JJ * 3) - 1 )
POS3 = ( (JJ * 3) - 0 )
```

```
C THIS IS SECTION ONE.  100 POINTS ARE COMPUTED. THEN ON TO MORE SECTIONS.
      DO N = 1, POINTS
P(N) = X(N) + (P1(N) * (A(POS2))) + (P2(N) * (A(POS3)))
Y(N,1) = P(N)* (B(POS1)) + P1(N) * (B(POS2)) + P2(N) * (B(POS3))
P2(N+1) = P1(N)
P1(N+1) = P(N)
      END DO

C THIS HANDLE SECTION TWO AND AS MANY MORE SECTIONS ARE IN THE FILTER.

IF ( SECTIONS .GE. 2 ) THEN

  DO JJ = 2, SECTIONS
POS1 = ( (JJ * 3) - 2 )
POS2 = ( (JJ * 3) - 1 )
POS3 = ( (JJ * 3) - 0 )

P1(1) = 0.0
P2(1) = 0.0

      DO N = 1, POINTS
P(N) = Y(N,(JJ-1)) + (P1(N) * (A(POS2))) + (P2(N) * (A(POS3)))
Y(N,JJ) = P(N)* (B(POS1)) + P1(N)* (B(POS2)) + P2(N) * (B(POS3))
P2(N+1) = P1(N)
P1(N+1) = P(N)

      END DO
  END DO
END IF
C
C THE ROUND-OFF ERROR IS ACCOUNTED FOR IN THIS NEXT SECTION.
C THE SAME STUCTURE IS USED, BUT I ROUNDOFF ALL CALCULATIONS
C IN THIS IMPULSE RESPONSE.
JJ = 1
POS1 = ( (JJ * 3) - 2 )
POS2 = ( (JJ * 3) - 1 )
POS3 = ( (JJ * 3) - 0 )

      DO N = 1, POINTS
PQ(N) = X(N) + ROUNDQ( ( P1Q(N) * A(POS2) ), NUMBITS ) +
     &          ROUNDQ( ( P2Q(N) * A(POS3) ), NUMBITS )
YQ(N,1) = ROUNDQ( ( PQ(N)  * B(POS1) ), NUMBITS ) +
     &  ROUNDQ( ( P1Q(N) * B(POS2) ), NUMBITS ) +
     &   ROUNDQ( ( P2Q(N) * B(POS3) ), NUMBITS )
P2Q(N+1) = P1Q(N)
```

```
P1Q(N+1) = PQ(N)

      END DO



IF ( SECTIONS .GE. 2 ) THEN

  DO JJ = 2, SECTIONS

POS1 = ( (JJ * 3) - 2 )
POS2 = ( (JJ * 3) - 1 )
POS3 = ( (JJ * 3) - 0 )

P1(1) = 0.0
P2(1) = 0.0

      DO N = 1, POINTS
PQ(N) = YQ(N,(JJ-1)) + ROUNDQ( (P1Q(N) * A(POS2)), NUMBITS ) +
     &          ROUNDQ( ( P2Q(N) * A(POS3) ), NUMBITS )
YQ(N,JJ) = ROUNDQ( ( PQ(N)  * B(POS1) ), NUMBITS ) +
     &     ROUNDQ( ( P1Q(N) * B(POS2) ), NUMBITS ) +
     &              ROUNDQ( ( P2Q(N) * B(POS3) ), NUMBITS )
P2Q(N+1) = P1Q(N)
P1Q(N+1) = PQ(N)

      END DO
  END DO
END IF


C THIS IS A RECURSIVE MEAN ESTIMATOR.  THE ARRAY SHOULD CONVERGE TO THE
C MEAN OF THE ERROR.  A NEAT WAY TO ITERATIVELY ESTIMATE THE MEAN.
C THE OTHER WAY TO PROGRAM THIS IN IS TO ADD UP THE ERROR ARRAY AND THEN
C DIVIDE BY THE NUMBER OF POINTS, USED FOR AN ARRAY THAT'S ALREADY COM-
C PLETE.  I GOT THE RECURSIVE ESTIMATOR FROM AN ADAPTIVE FILTERS CLASS.
C NEAT UH!
C
MEAN(0) = 0.0

DO N = 1, POINTS
ERROR(N) = Y(N,SECTIONS) - YQ(N,SECTIONS)
MEAN(N) = MEAN(N-1) + (1.0/N) * ( ERROR(N) - MEAN(N-1) )
END DO
```

```
MEANSE = ( (MEAN(POINTS)) ** 2 )
WRITE(6,3)'THE MEAN SQUARE ERROR  IS =',MEANSE


DO N = 1, POINTS
SUMOFERRORSQ = SUMOFERRORSQ + ( (ERROR(N) - MEAN(POINTS))**2 )
END DO

NOISEPOWER = ( 1.0/(POINTS - 1.0) ) * (SUMOFERRORSQ)

WRITE(6,1)'THE POWER FROM THE ROUND-OFF NOISE IS THE FOLLOWING:'
WRITE(6,4)
WRITE(6,3)'--                    POWER IN ROUND OFF NOISE IS =',NOISEPOWER



C ---------------------------------------------------------------------
1 FORMAT(/,A,$)
3 FORMAT(A,E12.4)
4 FORMAT(/)
5 FORMAT(A,$)
10 FORMAT(A,2X,3F10.5)
98 RETURN
99 END
C--- END OF ROUND-OFF ROUTINE.


C ***********************************************************************
C ***********************************************************************
C
C THIS SUBROUTINE IS CALLED BY SUBROUTINE ROUNDOFF.
C THE PURPOSE OF THIS ROUTINE IS TO FIND THE IMPULSE RESPONSE OF A
C DIRECT FORM DIGITAL FILTER TO 100 SAMPLE POINTS.  THEN THE ROUNDED
C IMPULSE RESPONSE IS FOUND.   THIS INFORMATION WILL THEN FIND THE
C POWER IN ROUND-OFF AS APPLIED TO THE FILTER.
C
C
SUBROUTINE ROUNDDIRECT(A,B,NX,NY,NUMBITS)

REAL Y(-200:200), X(-200:200)
REAL A(512), B(512), MEAN(0:100), MEANSE
REAL YQ(-200:200), ERROR(100), MEAN(100)
REAL SUMOFERRORQS, NOISEPOWER, XSUM, XQSUM, YSUM, YQSUM
BYTE ANS
INTEGER  NX, NY, POINTS
```

```
C
C I DO USE 100 POINTS.  HOWEVER, IF THE FILTER STRUCTURE IS JUST AN FIR,
C THEN ONLY THE NUMBER OF COEFFICIENTS ARE NECESSARY SINCE THE IMPULSE
C RESPONSE LASTS THAT LONG.
C HOWEVER, IF THE FILTER IS AN IIR STRUCTURE, THAN THE 100 POINTS IS
C REASONALBE TO VIEW THE IMPULSE RESPONSE.
C
C
C THE VALUES USED ARE ZEROED OUT.  THAT'S IN CASE A USER IS GOING THROUG
C THIS SECTION OF CODE FOR MULTIPLE RUNS.  I CAN'T HAVE VALUES LINGERING
C AROUND UPON ENTRANCE TO THIS CODE.
C
POINTS = 100
DO I = -200, POINTS
X(I)  = 0.0
Y(I)  = 0.0
YQ(I) = 0.0
END DO

X(1) = 1.0
XSUM = 0.0
YSUM = 0.0
XQSUM = 0.0
YQSUM = 0.0
MEAN(1) = 0.0
SUMOFERRORQS = 0.0

C ---------------------------------------------------------------
C
C A DIRECT FORM IS STRAIGHT FOWARD.  THE IMPULSE COMES IN AND THE OUTPOUT
C IS ADDED UP FROM EACH OF THE DELAY ELEMENTS.
C
DO N = 1, POINTS

DO J = 1, NX
XSUM = X(N-(J-1)) * B(J) + XSUM
END DO

DO J = 2, NY
YSUM = Y(N-(J-1)) * A(J) + YSUM
END DO

Y(N) = XSUM + YSUM
END DO
```

```
DO N = 1, POINTS
DO J = 1, NX
XQSUM = ROUNDQ( ( X(N-(J-1)) * B(J) + XQSUM ), NUMBITS )
END DO

DO J = 2, NY
YQSUM = ROUNDQ( ( YQ(N-(J-1)) * A(J) + YQSUM ), NUMBITS )
END DO

YQ(N) = XQSUM + YQSUM
END DO


C THIS IS A RECURSISE MEAN ESTIMATOR.  THE ARRAY SHOULD CONVERGE TO THE
C MEAN OF THE ERROR.  A NEAT WAY TO ITERATIVELY ESTIMATE THE MEAN.

MEAN(0) = 0.0
DO N = 1, POINTS
ERROR(N) = Y(N) - YQ(N)

MEAN(N) = MEAN(N-1) + (1.0/N) * ( ERROR(N) - MEAN(N-1) )

END DO

MEANSE = ( (MEAN(POINTS)) ** 2 )
WRITE(6,3)'THE MEAN SQUARE ERROR  IS =',MEANSE

DO N = 1, POINTS
SUMOFERRORSQ = SUMOFERRORSQ + ( (ERROR(N) - MEAN(POINTS))**2 )
END DO

NOISEPOWER = ( 1.0/(POINTS - 1.0) ) * (SUMOFERRORSQ)


WRITE(6,1)'THE POWER FROM THE ROUND-OFF NOISE IS THE FOLLOWING.'
WRITE(6,4)
WRITE(6,3)'---------- POWER IN ROUND OFF NOISE IS =',NOISEPOWER
WRITE(6,1)'   .............HIT RETURN KEY..........'
READ(5,5)ANS



1 FORMAT(/,A,$)
2 FORMAT(A,E10.3,2X,A,E10.3,A,E10.2,2X,A,E10.2)
```

```
3 FORMAT(A,E12.4)
4 FORMAT(/)
5 FORMAT(A,$)

98 RETURN
99 END
C--- END OF ROUND-OFF ROUTINE.


C ***********************************************************************
C ***********************************************************************
C
C THIS ROUTINE WILL CHANGE THE ORDER OF THE CASCADE SECTIONS.
C THE USER WILL ENTER TO NEW ORDER TO FOLLOW.  THE METHOD TO ENTER
C THE NEW COEFFICIENTS IS A SOURCE AND DESTINATION CASCADE SECTION
C NUMBER.  THE ROOTS OF EACH SECTION BOTH NUMERATOR AND THE DENOMENATOR
C IS FOUND AND SENT TO THE SCREEN AS WELL AS THE CASCADE SECTION'S
C COEFFICIENTS.  THIS WAY A USER CAN SEE THE POLE/ZERO LOCATIONS AND TELL
C IF THE MATCHING IS GOOD.  REMEMBER TO TRY TO MATCH THE CLOSEST POLE/ZERO
C PAIRS TOGETHER FOR THE OPTIMAL NOISE GENERATION. (MINI-MAX PRINCIPLE)
C
C COPY THE COEFFICIENTS INTO A TEMPORARY ARRARY FOR NON-DESTRUCTIVE
C TESTING.
C CHANGE SIGN ON POLES TO HAVE ALL PLUS SIGNS IN POLE SECTIONS.
C PRINTOUT THE COEFFICIENTS.
C FIND THE ROOTS TO THE POLYNOMIALS IN BOTH ZEROS AND POLES.
C (THAT MUST FIND THE ROOTS THAT ARE COMPLEX!)
C PRINT OUT THE COMPLEX ROOTS TO ALL THE CASCADE SECTIONS.
C BEGIN THE MANIPULATION OF THE SECTIONS BY MOVING SECOND ORDER SECTIONS
C IN THE NUMERATOR OR BY MOVING SECOND ORDER SECTIONS IN THE DENOMENATOR.
C RECALCULATE THE ROUNDOFF POWER ERROR IN THE NEW FILTER LAYOUT.
C CONTINUE ROUTINE IF USER DESIRES TO CONTINUE.
C


SUBROUTINE CHANGECASCADEORDER(A,B,NX,NY,NUMBITS)

COMPLEX  DENY, ROOTY(60), NUMX, ROOTX(60)
REAL A(512), B(512), AHOLD(512), BHOLD(512), THOLD1, THOLD2, THOLD3
REAL RADICAL, TESTIMAGINARY
BYTE ANS
INTEGER  NX, NY, NUMBITS, COUNT, ICONTINUEFLAG
INTEGER XSECTION, YSECTION, POS, SOURCEFROM, SOURCETO

C
```

```
C SAVE THE ORIGINAL DATA VALUES; NON-DESTRUCTIVE TESTING.
C
DO I = 1, NX
BHOLD(I) = B(I)
END DO

DO I = 1, NY
AHOLD(I) = A(I)
END DO


XSECTION = NX/3
YSECTION = NY/3


WRITE(6,1)'-.-.-.-.-.-.-.-.-.-TO ABORT ROUTINE, HIT RETURN-.-.-.-.-.-.-.-'
WRITE(6,1)'THIS ROUTINE WILL CHANGE THE ORDER OF THE CASCADE SECTIONS.'
WRITE(6,1)'YOU'LL BE ABLE TO MOVE A SINGLE CASCADE SECTION TO THE'
WRITE(6,1)'POSTION OF ANOTHER SINGLE CASCADE SECTION (SOURCE - TARGET).'


C*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*
C BEGIN THE DO WHILE SECTION
ICONTINUEFLAG = 1
DO WHILE (ICONTINUEFLAG .EQ. 1)
C
C I DO CHANGE THE SIGN SINCE THIS ROUTINE WILL PRINT OUT THE COEFFICIENTS
C TO THE SCREEN.  REMEMBER THE DEFINITION REQUIRES THAT THE COEFFICIENTS
C FOR THE POLE ARE INPUTED AS A NEGATIVE VALUE.  NOW THEY'LL BE PRINTED
C AS ALL PLUS SIGNS IN THE ZERO SECTIONS.  SEE THE HELP UTILITY FOR
C FURTHER EXPLAINATIONS.
C
C CHANGE THE NEGATIVE SIGN FOR THE DENOMENATOR TERM SO THE COEFFICIENTS
C ARE ALL POSITIVE COEFFICIENTS ( FROM THE DEFINITION OF CASCADE).
C RESTORE SIGN AT THE END OF THE ROUTINE.
C
C THIS IS A LONG ROUTINE.  IT COULD BE SPLIT UP INTO MUCH SMALL UNITS.
C

DO I = 0, (XSECTION - 1)

AHOLD(I*3+1) = 1.0  * AHOLD(I*3+1)
AHOLD(I*3+2) = -1.0 * AHOLD(I*3+2)
AHOLD(I*3+3) = -1.0 * AHOLD(I*3+3)
```

```
END DO



C
C PRINT OUT THE EXISTING COEFFICIENT FILE.
C

C ---------------------------------------------------------------------
C THIS SECTION WRITES OUT THE EXISTING INFORMATION TO THE SCREEN.
C
WRITE(6,4)
WRITE(6,11)'THE NUMBER OF SECTIONS IN THE NUMERATOR   =',XSECTION
WRITE(6,11)'THE NUMBER OF SECTIONS IN THE DENOMINATOR =',YSECTION

IF ( XSECTION .NE. YSECTION ) THEN
  WRITE(6,1)'MAKE EQUAL SECTIONS IN THE NUMERATOR AND DENOMINATOR.'
  WRITE(6,1)'USE ZERO'S AND A ONE "(1 - 0*Z^-1 - 0*Z^-2)" LIKE EXAMPLE.'
  READ(5,5)ANS
  GOTO 50
END IF

COUNT = 1

DO J = 0,( XSECTION - 1 )
WRITE(6,7)COUNT,COUNT
DO I = 1, 3
WRITE(6,25)( (J*3)+I ),BHOLD( (J*3)+I ),( (J*3)+I ),
     &                                  AHOLD( (J*3)+I )
END DO

COUNT = COUNT + 1
 END DO

C ---------------------------------------------------------------------
C THIS SECTION STARTS THE CASCADE SECTION MOVING.
C THE SECTIONS ARE SOURCE AND TARGET LOCATIONS FOR THE SECTION.
C

WRITE(6,1)'NOW WE'RE GOING TO LOOK AT THE ROOTS OF THE POLYNOMIALS.'


DO I = 0, (XSECTION - 1)
```

```
POS = (I * 3) + 1
C FIND OUT IF THE RADICAL IS GOING TO BE AN IMAGININARY NUMBER.
C FIND ROOTS FOR THE NUMERATOR SECTIONS
TESTIMAGINARY = (BHOLD(POS+1) )**2 - (4.0 * BHOLD(POS)*BHOLD(POS+2))

IF ( TESTIMAGINARY .LT. 0.0 ) THEN
RADICAL = SQRT( ABS(TESTIMAGINARY) )

C THESE ROOTS ARE IMAGINARY VALUED ROOTS
NUMX = CMPLX(-BHOLD(POS+1), RADICAL)
ROOTX(I*2+1) = NUMX/( 2.0 * BHOLD(POS) )

NUMX = CMPLX(-BHOLD(POS+1), -RADICAL)
ROOTX(I*2+2) = NUMX/( 2.0 * BHOLD(POS) )

ELSE

C THESE ROOTS ARE REAL VALUED ROOTS
NUMX = CMPLX(-BHOLD(POS+1) + RADICAL, 0.0)
ROOTX(I*2+1) = NUMX/( 2.0 * BHOLD(POS) )

NUMX = CMPLX(-BHOLD(POS+1) - RADICAL, 0.0)
ROOTX(I*2+2) = NUMX/( 2.0 * BHOLD(POS) )


END IF


TESTIMAGINARY = (A(POS+1) )**2 - (4.0 * A(POS)*A(POS+2))

C FIND ROOTS FOR THE DENOMENATOR SECTION
IF ( TESTIMAGINAY .LT. 0.0 ) THEN
RADICAL = SQRT( ABS(TESTIMAGINARY) )

DENY = CMPLX(-AHOLD(POS+1), RADICAL)
ROOTY(I*2+1) = DENY/( 2.0 * AHOLD(POS) )

DENY = CMPLX(-AHOLD(POS+1), -RADICAL)
ROOTY(I*2+2) = DENY/( 2.0 * AHOLD(POS) )

ELSE

DENY = CMPLX(-AHOLD(POS+1) + RADICAL, 0.0)
ROOTY(I*2+1) = DENY/( 2.0 * AHOLD(POS) )
```

```
DENY = CMPLX(-AHOLD(POS+1) - RADICAL, 0.0)
ROOTY(I*2+2) = DENY/( 2.0 * AHOLD(POS) )

END IF



END DO



C PRINT OUT THE ROOTS TO THE CASCADE SECTIONS.
COUNT = 1

DO J = 0,( XSECTION - 1 )


WRITE(6,7)COUNT,COUNT
WRITE(6,8)

   WRITE(6,26) (J*2+1),REAL(ROOTX(J*2+1)),AIMAG(ROOTX(J*2+1)),
     &         REAL(ROOTY(J*2+1)),AIMAG(ROOTY(J*2+1))


   WRITE(6,26) (J*2+2),REAL(ROOTX(J*2+2)),AIMAG(ROOTX(J*2+2)),
     &           REAL(ROOTY(J*2+2)),AIMAG(ROOTY(J*2+2))
COUNT = COUNT + 1


   END DO

C
C READY TO START THE MANIPULATIONS OF THE ROOTS FROM THE SECOND
C ORDER CASCADE SECTIONS.
C

WRITE(6,1)'DO YOU WANT TO MOVE SECTIONS IN THE NUMERATOR? [n]...:'

WRITE(6,1)'DO YOU WANT TO MOVE SECTIONS IN THE DENOMINATOR[d]...:    '
READ(5,5)ANS
IF ( ANS .EQ. ' ' ) THEN
GOTO 50
END IF
```

```
C
C____________DENOMINATOR SECTION_____________________________________________
C
IF ( (ANS .EQ. 'D') .OR. (ANS .EQ. 'd') ) THEN

30 WRITE(6,1)'ENTER NUMBER OF THE DENOMINATOR SECTION TO MOVE FROM ...:'
READ(5,10,END=50,ERR=30)SOURCEFROM
   IF ( SOURCEFROM .EQ. 0 ) THEN
GOTO 50
   END IF
IF ( (SOURCEFROM .LT. 1) .OR. (SOURCEFROM .GT. YSECTION) ) THEN
WRITE(6,1)'TRY AGAIN, BE CAREFUL!!!!!!!!!!!!'
GO TO 30
END IF

31 WRITE(6,1)'ENTER NUMBER OF THE DENOMINATOR SECTION TO MOVE TO ...:'
READ(5,10,END=50,ERR=31)SOURCETO
   IF ( SOURCETO .EQ. 0 ) THEN
GOTO 50
   END IF
IF ( (SOURCETO .LT. 1) .OR. (SOURCETO .GT. YSECTION) ) THEN
WRITE(6,1)'TRY AGAIN, BE CAREFUL!!!!!!!!!!!!'
GO TO 31
END IF

THOLD1 = AHOLD( (SOURCETO * 3) - 2)
THOLD2 = AHOLD( (SOURCETO * 3) - 1)
THOLD3 = AHOLD( (SOURCETO * 3) - 0)

AHOLD( (SOURCETO * 3) - 2) = AHOLD( (SOURCEFROM * 3) -2)
AHOLD( (SOURCETO * 3) - 1) = AHOLD( (SOURCEFROM * 3) -1)
AHOLD( (SOURCETO * 3) - 0) = AHOLD( (SOURCEFROM * 3) -0)

AHOLD( (SOURCEFROM * 3) -2) = THOLD1
AHOLD( (SOURCEFROM * 3) -1) = THOLD2
AHOLD( (SOURCEFROM * 3) -0) = THOLD3

END IF
C
C________NUMERATOR SECTION____________________________________________________
C

IF ( (ANS .EQ. 'N') .OR. (ANS .EQ. 'n') ) THEN

32 WRITE(6,1)'ENTER THE NUMBER OF THE NUMERATOR SECTION TO MOVE FROM ...:'
```

```
READ(5,10,END=50,ERR=32)SOURCEFROM

IF ( (SOURCEFROM .LT. 1) .OR. (SOURCEFROM .GT. XSECTION) ) THEN
WRITE(6,1)'TRY AGAIN, BE CAREFUL!!!!!!!!!!!!'
GO TO 32
END IF

33 WRITE(6,1)'ENTER THE NUMBER OF THE NUMERATOR SECTION TO MOVE TO ...:'
READ(5,10,END=50,ERR=33)SOURCETO

IF ( (SOURCETO .LT. 1) .OR. (SOURCETO .GT. XSECTION) ) THEN
WRITE(6,1)'TRY AGAIN, BE CAREFUL!!!!!!!!!!!!'
GO TO 33
END IF


THOLD1 = BHOLD( (SOURCETO * 3) - 2)
THOLD2 = BHOLD( (SOURCETO * 3) - 1)
THOLD3 = BHOLD( (SOURCETO * 3) - 0)

BHOLD( (SOURCETO * 3) - 2) = BHOLD( (SOURCEFROM * 3) -2)
BHOLD( (SOURCETO * 3) - 1) = BHOLD( (SOURCEFROM * 3) -1)
BHOLD( (SOURCETO * 3) - 0) = BHOLD( (SOURCEFROM * 3) -0)

BHOLD( (SOURCEFROM * 3) -2) = THOLD1
BHOLD( (SOURCEFROM * 3) -1) = THOLD2
BHOLD( (SOURCEFROM * 3) -0) = THOLD3

END IF
C
C ---------------------------------------------------------------

C ---------------------------------------------------------------
C
C NOW RUN THE ROUNDOFF POWER ERROR ROUTINE WITH THESE COEFFICIENTS
C


CALL ROUNDCASCADE(A,B,NX,NY,NUMBITS)


WRITE(6,1)'I WANT TO CONTINUE TO MANUVER CASCADE SECTIONS? [Y]es  ...:'
READ(5,5)ANS

IF ( (ANS .EQ. 'Y') .OR. (ANS .EQ. 'y') ) THEN
```

```
ICONTINUEFLAG = 1

ELSE

ICONTINUEFLAG = 0

END IF

END DO

C*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*!*
C END THE DO WHILE SECTION


C ON ERRORS, GO TO THE END OF THE ROUTINE.
C THE COEFFICIENTS ARE NOT SAVED SINCE THIS IS A NON-DESTRUCTIVE
C ROUND-OFF POWER CALCULATION ROUTINE.

50 CONTINUE


1 FORMAT(/,A,$)
2 FORMAT(A,I2,3X,A,I2)
3 FORMAT(A,E12.4)
4 FORMAT(/)
5 FORMAT(A,$)
7 FORMAT(/,/,'   FROM NUMERATOR SECTION (',I2,')     AND ',
     &           'DENOMINATOR SECTION (',I2,')')
8 FORMAT(/,'         REAL-PART    IMAGINARY-PART
     &REAL-PART     IMAGINARY-PART')
10 FORMAT(I,$)
11 FORMAT(A,I3)
12 format(f12.7)
15 FORMAT(A,I3,5X,A,I3)
20 FORMAT(2X,'A(',I3,') = ',F11.7)
21 FORMAT(2X,'B(',I3,') = ',F11.7)
25 FORMAT(2X,'B(',I3,') = ',F11.7,4X,'A(',I3,') = ',F11.7)
26 FORMAT('*ROOT(',I2,')',E12.5,2X,E12.5,2X,E12.5,2X,E12.5)

98 RETURN
99 END
C--- END OF ROUND-OFF ROUTINE.
```

```
C ***************************************************************************
C ***************************************************************************
c THIS SUBROUTINE WILL READ IN THE COEFFICIENTS FOR THE FILTER DESIRED
C FROM THE DISK FILE SYSTEM OR A USER CAN USE THE KEYBOARD TO INPUT
C COEFFICIENTS.
C
SUBROUTINE READCOEFF(NX,NY,A,B,FNAME,ANS,FILTERTYPE)

CHARACTER*15 FNAME,TNAME
BYTE ANS,HFLAG
REAL A(512),B(512)
INTEGER NX,NY, XSECTION, YSECTION, FILTERTYPE,COUNT

41 HFLAG = 'F'
TNAME = FNAME
WRITE(6,1) 'DIFFERENCE EQUATION COEFFICIENTS FROM FILE?     [RET] :'
WRITE(6,1) 'DIFFERENCE EQUATION COEFFICIENTS FROM KEYBOARD  [Y]   :'
READ(5,5,END=99,ERR=41)ANS

C ***************************************************************************

  IF ( ANS .EQ. 'y' ) THEN
ANS = 'Y'
ENDIF

C ***************************************************************************

  IF ( ANS .EQ. ' ' ) THEN

C--- ---READ THE COEFFICIENTS FROM THE HOME DISK DIRECTORY---

WRITE(6,1)' ***  BE SURE COEFFICIENTS ARE <=+-1.0 AND INCLUDE DECIMAL. ***'
WRITE(6,1)' ***  THERE IS A NORMALIZATION ROUTINE IF YOU NEED IT     ***'
WRITE(6,4)
WRITE(6,17)FNAME
WRITE(6,1)'*ENTER NAME OF COEFFICIENT FILE*:   [NAME] :'
WRITE(6,1)'*ENTER RETURN FOR NOCHANGE*:        [RET]  :   '
 READ(5,5,END=99,ERR=41) FNAME

 IF ( FNAME .EQ. ' ' ) THEN
FNAME = TNAME
 ENDIF

WRITE(6,4)
WRITE(6,1)'THE INPUT FILE WITH THE COEFFICIENTS IS BEING READ IN'
```

```
WRITE(6,17)FNAME

   OPEN(4,NAME=FNAME,STATUS='OLD',ERR=41)
   READ(4,2,END=99,ERR=98)NX
   READ(4,2,END=99,ERR=98)NY

    DO J = 1,NX
READ(4,3,END=99,ERR=98)B(J)

  IF ( ABS(B(J)) .GT. 1.0 ) THEN
HFLAG = 'T'
  END IF

  END DO


    DO J = 1,NY
READ(4,3,END=99,ERR=98)A(J)

  IF ( ABS(A(J)) .GT. 1.0 ) THEN
HFLAG = 'T'
  END IF

    END DO

    CLOSE(4)

C -------------------------------------------------------

C NOW PRINT OUT THE COEFFICIENTS TO THE SCREEN.
C FILTERTYPE 2 IS THE CASCASDE FORM.
C THE TYPE IS DIRECT FORM IS PRINTED.

IF ( FILTERTYPE .EQ. 1 ) THEN

DO J = 1, NX
WRITE(6,21)J-1,B(J)
END DO

DO J = 1, NY
WRITE(6,20)J-1,A(J)
END DO

ELSE
```

```
CONTINUE

END IF

C
C NOW THE CASCADE FORM IS PRINTED OUT TO THE SCREEN.
C -----------------------------------------------------

IF ( FILTERTYPE .EQ. (2) ) THEN

XSECTION = NX/3
YSECTION = NY/3

C THIS SECTION WRITES OUT THE EXISTING INFORMATION TO THE SCREEN.

WRITE(6,4)
WRITE(6,8)'THE NUMBER OF SECTIONS IN THE NUMERATOR   =',XSECTION
WRITE(6,8)'THE NUMBER OF SECTIONS IN THE DENOMINATOR =',YSECTION

IF ( XSECTION .NE. YSECTION ) THEN
  WRITE(6,1)'MAKE EQUAL SECTIONS IN THE NUMERATOR AND DENOMINATOR.'
  WRITE(6,1)'USE ZERO'S AND A ONE "(1 - 0*Z^-1 - 0*Z^-2)" LIKE EXAMPLE.'
  READ(5,5)ANS
  RETURN
ELSE

CONTINUE

END IF

COUN1 = 1

DO J = 0,( XSECTION - 1 )
WRITE(6,7)COUNT,COUNT

DO I = 1, 3
WRITE(6,25)( (J*3)+I ),B( (J*3)+I ),( (J*3)+I ),A( (J*3)+I )
END DO

COUNT = COUNT + 1

 END DO

END IF
C -----------------------------------------------------
```

```
END IF


WRITE(6,4)
C
C THE USER WILL ENTER THE COEFFICIENTS FROM THE KEYBOARD.
C
  IF ( ANS .EQ. 'Y' ) THEN


C--- ---READ THE COEFFICIENTS FROM KEYBOARD---

43   WRITE(6,1) '  ENTER NUMBER OF TERMS IN X(n): '
  READ(5,2,END=99,ERR=43) NX

   DO J = 1,NX
37      WRITE(6,11)J-1
     READ(5,3,END=99,ERR=43)B(J)

  IF ( B(J) .GT. 1.0 ) THEN
HFLAG = 'T'
  END IF

   END DO

44   WRITE(6,4)

  WRITE(6,1) '  ENTER NUMBER OF TERMS IN Y(n): '
  READ(5,2,END=99,ERR=44) NY

   DO J = 1,NY
     WRITE(6,10)J-1
     READ(5,3,END=99,ERR=44)A(J)

  IF ( A(J) .GT. 1.0 ) THEN
HFLAG = 'T'
  END IF

      END DO

C SINCE THE COEFFICIENTS WERE ENTERED IN FROM THE KEYBOARD, THEY NEED
C TO BE SAVED TO THE DISK DRIVE.

CALL SAVECOEF(FNAME,NX,NY,A,B)
```

```
ENDIF

C--- THIS IS A TEST FOR COEFFICIENTS THAT ARE GREATER THAN ONE.

 IF ( HFLAG .EQ. 'T' ) THEN

WRITE(6,1)'** MAGNITUDE OF COEFFICIENTS  GREATER THAN ONE **!!!'
WRITE(6,1)'YOU MUST USE THE NORMALIZATION ROUTINE!!!!!!!!!!!!'
WRITE(6,1)'**************     SELECTION #8     ***************'
END IF

WRITE(6,1)'HIT RETURN KEY TO CONTINUE  [RETURN KEY]: '
READ(5,2,END=99,ERR=98)ANS
WRITE(6,4)



1 FORMAT(/,A,$)
2 FORMAT(I3)
3 FORMAT(E14.7)
4 FORMAT(/)
5 FORMAT(A,$)
7 FORMAT(/,'NUMERATOR SECTION (',I2,')',3X,'DENOMINATOR SECTION (',I2,')')
8 FORMAT(A,I3)
10 FORMAT(2X,'A(',I1,')= ',$)
11 FORMAT(2X,'B(',I1,')= ',$)
17 FORMAT(/,'CURRENT NAME OF THE COEFFICIENT FILE IS "',A,'"',/)
20 FORMAT(2X,'A(',I3,') = ',E16.8)
21 FORMAT(2X,'B(',I3,') = ',E16.8)
25 FORMAT(2X,'B(',I3,') = ',F11.7,4X,'A(',I3,') = ',F11.7)

98 RETURN
99 END
C--- END OF INPUT ROUTINE, THE COEFFICIENTS HAVE BEEN READ IN.
C ***********************************************************************



C ***********************************************************************
C ***********************************************************************
C THIS SUBROUTINE ALLOWS THE USER TO DETERMINE THE NUMBER OF BITS
C TO USE FOR THE SIMULATION CALCULATIONS.
C
SUBROUTINE BITSAVAILABLE(NUMBITS)
```

```
INTEGER NUMBITS,TNUMBITS

TNUMBITS = NUMBITS
41 WRITE(6,4)
write(6,1) '  Enter the NUMBER of Bits for Quantizing:  '
WRITE(6,1) '  ENTER [RET] FOR NO CHANGE              :  '
READ(5,2,END=99,ERR=41) NUMBITS

IF ( NUMBITS .EQ. (0) ) THEN
NUMBITS = TNUMBITS
END IF

IF ( NUMBITS .GT. (32) ) THEN
WRITE(6,1) '** OUT ** OF ** BOUNDS **'
NUMBITS = TNUMBITS
END IF

IF ( NUMBITS .LT. (1) ) THEN
WRITE(6,1) '** OUT ** OF ** BOUNDS **'
NUMBITS = TNUMBITS
END IF


WRITE(6,5)NUMBITS
1 FORMAT(/,A,$)
2 FORMAT(I)
4 FORMAT(//)
5 FORMAT(/,'THE NUMBER OF BITS IS =',I3)
RETURN
99 END

C ***********************************************************************
C ***********************************************************************
C FILENUMBER IS A TWO DIGIT NUMBER FOR THE USER TO ASSCOCIATE WITH THE
C OUTPUT FILES.  THE OUTPUT FILES INCLUDE THE QUANTIZED MAG/PHASE AND THE
C ERROR DATA FILE.  THE UNQUANTIZED PLOTS ARE DONE IN THE NEXT SUBROUTINE.

SUBROUTINE FILENUMBER(FNUMBER)
CHARACTER*2 FNUMBER,TFNUMBER

71 WRITE(6,4)
TFNUMBER = FNUMBER
WRITE(6,1)'ENTER THE NUMBER TO ASSOCIATE WITH THE OUTPUT FILES:'
WRITE(6,1) '  ENTER THE NUMBER AS A **TWO** DIGIT NUMBER     [##] :'
WRITE(6,1) '  ENTER [RET] FOR NO CHANGE                   [RET]:'
```

```
READ(5,5,END=99,ERR=71) FNUMBER

 IF ( FNUMBER .EQ. ' ' ) THEN
FNUMBER = TFNUMBER
END IF

WRITE(6,4)
WRITE(6,6)' *****   THE FILE NUMBER YOU HAVE CHOSEN IS:',FNUMBER
WRITE(6,4)
1 FORMAT(/,A,$)
4 FORMAT(/,/)
5 FORMAT(A)
6 FORMAT(A,A,/,/,/)
98 RETURN
99 END
C ***************************************************************************
C THE UNQNUMFILE IS A TWO DIGIT NUMBER (COULD BE ASCII) THAT A USER INPUTS
C FOR THE SOFTWARE TO TAG ON THE UNQUANTIZED MAGNITUDE AND PHASE PLOTS.
C
SUBROUTINE UNQNUMFILE(UNQFNAME)
CHARACTER*2 UNQFNAME,TNAME

WRITE(6,4)
TNAME = UNQFNAME
31 WRITE(6,1)'ENTER NUMBER FOR THE UNQUANTIZED MAG/PHASE FILES:'
WRITE(6,1) '  ENTER THE NUMBER AS A **TWO** DIGIT NUMBER     [##] :'
WRITE(6,1) '  ENTER [RET] FOR NO CHANGE                    [RET]:'

READ(5,5,END=99,ERR=31) UNQFNAME

 IF ( UNQFNAME .EQ. ' ' ) THEN
UNQFNAME = TNAME
END IF
WRITE(6,4)
WRITE(6,6)'THE NUMBER YOU HAVE CHOSEN FOR UNQUANTIZED FILES IS:',UNQFNAME
WRITE(6,4)
1 FORMAT(/,A,$)
4 FORMAT(/,/)
5 FORMAT(A)
6 FORMAT(A,A,/,/,/)
RETURN
99 END
C ***************************************************************************
```

```
C ************************************************************************
C THIS SUBROUTINE SETS UP THE FILTERTYPE WITH A NUMBER THAT TELLS ME
C WHAT TYPE OF FILTER IS BEING EVALUATED.  TWO TYPES OF FILTERS CAN BE
C SIMULATED, A DIRECT FORM OR A CASCADE FORM FILTER.
C
C ALSO,
C THIS ROUTINE DOES A CONVERSION PROCESS.
C A USER CAN CONVERT FROM A CASCADE ORGANIZATION TO A DIRECT FORM FILTER.
C THIS PROCESS OF CONVERSION MULITPLIES OUT THE POLYNONIALS (SECOND ORDER
C SECTIONS) INTO ONE POLYNOMIAL FOR THE POLES AN ZEROS.
C
C
SUBROUTINE TYPEOFFILTER(FILTERTYPE,FILTERNAME,FNAME,NX,NY,A,B)
INTEGER FILTERTYPE, THOLD, NX, NY, IANS
REAL A(512),B(512)
CHARACTER*15 FILTERNAME,FNAME


THOLD = FILTERTYPE

32 WRITE(6,1)'              ***** DO YOU WANT TO *****                    '
WRITE(6,1)'(1) ANALYZE --CASCADE-- FORM, OR                    [1]   :'
WRITE(6,1)'(2) ANALYZE --DIRECT--  FORM              [RET] or [2]   :'
WRITE(6,1)'(3) ENTER CONVERSION PROCESS FROM CASCADE TO DIRECT [3]   :'
WRITE(6,1)'                                                  .....:'
READ(5,5,END=99,ERR=32)IANS


 IF ( IANS .EQ. '0' ) THEN
IANS = '2'
ENDIF



IF ( IANS .EQ. '1' ) THEN


FILTERTYPE = 2
FILTERNAME = 'CASCADE'
WRITE(6,1)'  BE SURE TO INPUT COEFFICIENTS LIKE IN EQ. 6.23 '
WRITE(6,1)'   FROM OPPENHEIM AND SCHAFER..................  '
WRITE(6,1)'  BUT INCLUDE THE '1.0' IN THE DENOMINATOR AS A COEFFICIENT'
WRITE(6,1)'  EACH SECTION WILL HAVE SIX COEFFICIENTS.. ..............'
WRITE(6,1)'         **************  EXAMPLE  ***************  '
```

```
WRITE(6,1)'             EACH SECTION IS LOOKS LIKE THIS     '
WRITE(6,1)' '
WRITE(6,1)'          (B_0 + B_1*Z^-1 + B_2*Z^-2)   (...)     '
WRITE(6,1)' H(Z) =   ------------------------- * ----- * ...'
WRITE(6,1)'          (A_0 - A_1*Z^-1 - A_2*Z^-2)   (...) '
WRITE(6,1)' '
WRITE(6,1)'                              note: postion A_0 = 1.0 always'


ELSE


 IF ( IANS .EQ. '2' ) THEN

FILTERTYPE = 1
FILTERNAME = 'DIRECT FORM'
WRITE(6,1)'THE X COEFFICIENTS ARE ON THE RIGHT SIDE OF DIFFERENCE EQUATION'
WRITE(6,1)'  THE Y COEFFICIENTS ARE ON THE LEFT SIDE OF DIFFERENCE EQUATION'

WRITE(6,1)'  BE SURE TO INPUT COEFFICIENTS LIKE IN EQ. 5.16 '
WRITE(6,1)'   FROM OPPENHEIM AND SCHAFER..................  '
WRITE(6,1)'   '
WRITE(6,1)'         ***************  EXAMPLE  ***************  '
WRITE(6,1)'             EACH FILTER LOOKS LIKE THIS     '
WRITE(6,1)' '
WRITE(6,1)'    Y(Z)*A_0 + Y(Z^-1)*A_1 + Y(Z^-2)*A_2 + ...   =   '
WRITE(6,1)'                                          '
WRITE(6,1)'    X(Z)*B_0 + X(Z^-1)*B_1 + X(Z^-2)*B_2 + ...       '
WRITE(6,1)' '
WRITE(6,1)'                                    '



ELSE

CONTINUE

ENDIF


ENDIF

C
C THE CONVERSION PROCESS IS A SUBROUTINE SINCE IT DOES A BIT OF
C CALCULATIONS.
```

```
C

IF ( IANS .EQ. '3' ) THEN
CALL CONVERSION(FILTERTYPE,FILTERNAME,FNAME,NX,NY,A,B)
ENDIF


WRITE(6,4)
WRITE(6,1)'ENTER ANY KEY TO CONTINUE  [RETURN KEY] ...:'
READ(5,5,END=99,ERR=98)ANS
WRITE(6,4)

1 FORMAT(/,A,$)
4 FORMAT(/,/,/)
5 FORMAT(A)
98 RETURN
99 END

C ***************************************************************************
C ***************************************************************************
C THIS SUBROUTINE IS CALLED BY SUBROUTINE TYPEOFFILTER.
C
C THIS ROUTINE DOES A CONVERSION PROCESS.
C A USER CAN CONVERT FROM A CASCADE ORGANIZATION TO A DIRECT FORM FILTER.
C THIS PROCESS OF CONVERSION MULITPLIES OUT THE POLYNONIALS (SECOND ORDER
C SECTIONS) INTO ONE POLYNOMIAL FOR THE POLES AN ZEROS.
C
C

SUBROUTINE CONVERSION(FILTERTYPE,FILTERNAME,FNAME,NX,NY,A,B)

INTEGER SECTIONS, COUNT, NX, NY, FILTERTYPE
REAL A(512), B(512), PX(-1:200), PY(-1:200), PXT(-1:200), PYT(-1:200)
CHARACTER*15 FNAME, FILTERNAME
BYTE ANS,HFLAG

IF (  (NX .EQ. 0)  ) THEN

 WRITE(6,1)'YOU MUST FIRST READ OR CREATE A COEFFICIENT FILE TO USE!!!'
 WRITE(6,1)'IT MUST FOLLOW THE CASCADE DESIGN OPPENHEIM/SHAFFER EQ.6.23'
 WRITE(6,1)'    RUN THE COEFFICIENT INPUT ROUTINE FOR FURTHER DETAILS'
 RETURN

END IF
```

```
SECTIONS = NX/3
COUNT = 0

C--- IF A USER STARTS WITH A LARGER FILE AND THEN TRIES A SMALLER FILE,
C--- SOME VALUES WILL BE LEFT OVER THAT NEED TO BE ZERO FOR THE ALGORITHM
C--- TO WORK.  THIS ROUTINE WILL REACH OUT INTO THE ARRAYS PX(),PY().

DO I = -1, ( NX )
PX( I ) = 0.0
END DO

DO I = -1, ( NY )
PY( I ) = 0.0
END DO


C--- THE FIRST LEVEL FOR CONVERSION IS DONE EXPLICITLY.
C--- WE ARE MULTIPLYING TWO SECOND ORDER SECTIONS
C--- LEVEL ONE
PXT(1) = B(1) * B(4) + B(2) * 0    + B(3) * 0
PXT(2) = B(1) * B(5) + B(2) * B(4) + B(3) * 0
PXT(3) = B(1) * B(6) + B(2) * B(5) + B(3) * B(4)
PXT(4) = B(1) * 0    + B(2) * B(6) + B(3) * B(5)
PXT(5) = B(1) * 0    + B(2) * 0    + B(3) * B(6)

DO I =1, 5

PX(I) = PXT(I)
END DO

write(6,2)sections


C-----------------------------------------------
IF ( SECTIONS .EQ. 2 ) THEN

CONTINUE

ELSE


 DO K=1,(SECTIONS - 2)
write(6,2)count
write(6,1)'this is the start of the do loops'
```

```
C--- THIS IS THE NEXT LEVEL COMPUTATION.  RESULTS FROM PREVIOUS LEVEL ARE
C--- NEEDED.  AFTER EACH LEVEL IS COMPUTED, THE RESULTS MUST BE FEED BACK
C--- INTO THE INPUT TERMS FOR THE NEXT LEVEL TO USE.

NUM = ( 3 * COUNT )


    DO J=1,(7+(2*COUNT))

write(6,2)j
write(6,22)px(j),px(j-1),px(j-2)
write(6,23)b(7+num),b(8+num),b(9+num)

        PXT(J)=B(7+NUM) * PX(J) + B(8+NUM)*PX(J-1) + B(9+NUM)*PX(J-2)

    END DO


C--- THIS RESTORES VALUES FOR NEXT LEVEL COMPUTATION.
c--- AND CLEARS THE ARRAY BEFORE A USER CAN REENTER THIS CODE.

DO N = 1, ( 7 + (2 * COUNT) )
PX(N) = PXT(N)
END DO

COUNT = COUNT + 1

 END DO


END IF
C-----------------------------------------------



c--- UPDATE THE VALUE OF NX HOW MANY TERMS ON RIGHT SIDE OF DIFF. EQ.

NX = ( SECTIONS * 2 + 1 )

C---
C--- NOW RESTORE THE "X" COEFFICIENTS SO THE PROGRAM CAN USE THESE VALUES!

DO J = 1, NX
B(J) = PX(J)
```

```
END DO



c---    ************************************************************************

IF (  (NY .EQ. 0)  ) THEN

WRITE(6,1)'YOU MUST FIRST READ OR CREATE A COEFFICIENT FILE TO USE!!'
WRITE(6,1)'IT MUST FOLLOW THE CASCADE DESIGN OPPENHEIM/SHAFFER EQ.6.23'
WRITE(6,1)'    RUN THE COEFFICIENT INPUT ROUTINE FOR FURTHER DETAILS  '
RETURN

END IF


SECTIONS = NY/3
COUNT = 0

C--- THE FIRST LEVEL FOR CONVERSION IS DONE EXPLICITLY.
C--- WE ARE MULTIPLYING TWO SECOND ORDER SECTIONS.

C--- Y COEFFICIENTS ARE  ( 1 - (A_1,K * Z^-1) - (A_2,K * Z^-2) )  IN
C--- THE DENOMINATOR AS IN OPPENHEIM/SHAFFER EQ. 6.23.
C--- THESE SECOND ORDER SECTIONS ARE CASCADED.  I CONVERT SIGN FIRST, SO I
C--- DON'T HAVE TO TRACK THE UNIQUE SIGN CHANGES WITHIN THE ALGORITHM.

DO I = 0, (SECTIONS - 1)

A(I * 3 + 1) = +A(I * 3 + 1)
A(I * 3 + 2) = -A(I * 3 + 2)
A(I * 3 + 3) = -A(I * 3 + 3)

END DO


C--- LEVEL ONE
PYT(1) = A(1) * A(4) + A(2) * 0    + A(3) * 0
PYT(2) = A(1) * A(5) + A(2) * A(4) + A(3) * 0
PYT(3) = A(1) * A(6) + A(2) * A(5) + A(3) * A(4)
PYT(4) = A(1) * 0    + A(2) * A(6) + A(3) * A(5)
PYT(5) = A(1) * 0    + A(2) * 0    + A(3) * A(6)

DO I = 1, 5
```

```
PY(I) = PYT(I)
END DO

C-------------------------------------------------------
IF ( SECTIONS .EQ. 2 ) THEN
CONTINUE
ELSE

 DO K=1,(SECTIONS - 2)


C--- THIS IS THE NEXT LEVEL COMPUTATION.  RESULTS FROM PREVIOUS LEVEL ARE
C--- NEEDED.  AFTER EACH LEVEL IS COMPUTED, THE RESULTS MUST BE FEED BACK
C--- INTO THE INPUT TERMS FOR THE NEXT LEVEL TO USE.

NUM = (3*COUNT)

    DO J=1,(7+(2*COUNT))

PYT(J)=A(7+NUM) * PY(J) + A(8+NUM)*PY(J-1) + A(9+NUM)*PY(J-2)

   END DO

C--- THIS RESTORES VALUES FOR NEXT LEVEL COMPUTATION
C--- AND CLEARS THE ARRAY BEFORE A USER RE-ENTERS THIS CODE.

DO N = 1, ( 7 + (2 * COUNT) )
PY(N) = PYT(N)
END DO


  COUNT = COUNT + 1

 END DO

END IF
C-------------------------------------------------------

c--- UPDATE THE VALUE OF NX HOW MANY TERMS ON RIGHT SIDE OF DIFF. EQ.

NY = ( SECTIONS * 2 + 1 )

C---
C--- NOW RESTORE THE "Y" COEFFICIENTS SO THE PROGRAM CAN USE THESE VALUES!
```

```
DO J = 1, NY
A(J) = PY(J)
END DO



WRITE(6,4)
41 WRITE(6,1) '  *******CONVERSION UTILITY: CASCADE TO DIRECT FORM*******'
WRITE(6,1) '  ***    THIS ROUTINE WILL STORE YOUR COEFFICIENTS     ***'
WRITE(6,1) '  ***          TO THE FILE THAT YOU ENTER.             ***'
WRITE(6,1) '  ***    YOU WILL HAVE COEFFICIENTS IN DIRECT FORM!    ***'
WRITE(6,1) '..........ENTER FILE NAME TO STORE COEFFICIENTS.........:'

READ(5,5,END=99,ERR=41)FNAME

WRITE(6,6)'THE FILE NAME IS CALLED:',FNAME
WRITE(6,1)'SHALL I PROCEED TO CONVERT AND SAVE COEFFICIENTS?  ENTER[Y]:'
WRITE(6,1)'WAIT, START THIS ROUTINE AGAIN?                    [RETURN KEY]:'
READ(5,5,END=99,ERR=41)ANS
  IF ( (ANS .EQ. 'Y') .OR. (ANS .EQ. 'y') ) THEN
CONTINUE
ELSE
GOTO 41
ENDIF

FILTERTYPE = 1
FILTERNAME = 'DIRECT FORM'

WRITE(6,1)' THE X COEFFICIENTS ARE ON THE RIGHT SIDE OF DIFFERENCE EQUATION'
WRITE(6,1)'  THE Y COEFFICIENTS ARE ON THE LEFT SIDE OF DIFFERENCE EQUATION'

C THE HFLAG TELLS ME IF THE COEFFICIENTS NEED NORMALIZATION.
HFLAG = 'F'

OPEN(10,NAME=FNAME,STATUS='UNKNOWN',ERR=98)
WRITE(10,8)NX
WRITE(10,8)NY
WRITE(6,4)
DO I = 1,NX
WRITE(10,3)B(I)
WRITE(6,21)I-1,B(I)
IF ( B(I) .GT. (1.0) ) THEN
HFLAG = 'T'
END IF
END DO
```

```
WRITE(6,7)
DO I = 1,NY
WRITE(6,20)I-1,A(I)
WRITE(10,3)A(I)
IF ( A(I) .GT. (1.0) ) THEN
HFLAG = 'T'
END IF
END DO

CLOSE(10)


 IF ( HFLAG .EQ. 'T' ) THEN

WRITE(6,1)'** MAGNITUDE OF COEFFICIENTS  GREATER THAN ONE **!!!'
WRITE(6,1)'YOU MUST USE THE NORMALIZATION ROUTINE!!!!!!!!!!!!'
WRITE(6,1)'***************     SELECTION #8     ***************'
END IF

WRITE(6,1)'HIT RETURN KEY TO CONTINUE  [RETURN KEY]: '
READ(5,2,END=99,ERR=98)ANS
WRITE(6,4)



WRITE(6,7)
WRITE(6,1) 'SAVING COEFFICIENTS COMPLETE; COEFFICIENTS WRITTEN.'
WRITE(6,7)
WRITE(6,6) 'THE NAME OF YOUR COEFFICIENT FILE IS :',FNAME
WRITE(6,4)


1 FORMAT(/,A,$)
2 FORMAT(I)
3 FORMAT(F15.10)
4 FORMAT(/,/,/)
5 FORMAT(A,$)
6 FORMAT(/,A,A)
7 FORMAT(/)
8 FORMAT(I3)
10 FORMAT('THE NX IS=',I)
11 FORMAT('THE NY IS=',I`
12 FORMAT('THE VALUE IS',:15.5)
20 FORMAT(2X,'A(',I3,') = ',F21 10)
21 FORMAT(2X,'B(',I3,') = ',F21.10)
```

```
22 FORMAT(/,' PX(J) =   ',F7.1,' PX(J-1) = ',F7.1,' PX(J-2) = ',F7.1)
23 FORMAT(/,' B(7+NUM) =',F7.1,' B(8+NUM) =',F7.1,' B(9+NUM) =',F7.1)

98 RETURN
99 END



C ***************************************************************************
C ***************************************************************************
C THIS SUBROUTINE GETS THE NUMBER OF SPECTRAL POINTS TO EVALUATE.
C THE OUTPUT FILES WILL ALSO HAVE THIS MANY DATA POINTS IN EACH OF THEM.
C
SUBROUTINE ITERATIONS(STEP,NPOINTS,RANGE)

INTEGER NPOINTS,TNPOINTS
REAL STEP, RANGE
BYTE DUMMY

TNPOINTS = NPOINTS

39 WRITE(6,1) '  ENTER NUMBER OF SPECTRAL POINTS TO EVALUATE: '
WRITE(6,1) '  ENTER  [RET] FOR NOCHANGE                    : '
READ(5,2,END=99,ERR=39) NPOINTS
WRITE(6,4)

 IF ( NPOINTS .EQ. (0) ) THEN
NPOINTS = TNPOINTS
END IF

IF ( NPOINTS .GT. 2950 ) THEN

WRITE(6,1)'**** TOO LARGE FOR SYSTEM TO HANDLE ***(GET BIGGER MACHINE)'
WRITE(6,1)'------------HIT RETURN KEY TO CONTINUE---------'
READ(5,7,END=99,ERR=39)DUMMY
NPOINTS = TNPOINTS

END IF


IF ( NPOINTS .LT. 1) THEN

WRITE(6,1)'TRY AGAIN!!!!!!!!'
NPOINTS = TNPOINTS
WRITE(6,1)'------ -----HIT RETURN KEY TO CONTINUE---------'
```

```
READ(5,7,END=99,ERR=39)DUMMY

END IF

STEP = RANGE / NPOINTS

WRITE(6,3)NPOINTS
WRITE(6,4)
1 FORMAT(/,A,$)
2 FORMAT(I)
3 FORMAT(/,'THE NUMBER OF SPECTRAL POINTS IS SET AT "',I4,'"')
4 FORMAT(/)
5 FORMAT(/)
7 FORMAT(A)
RETURN
99 END

C ***************************************************************************
C ***************************************************************************
C THIS ROUTINE WILL ALLOW A USER TO FOCUS IN ON A SPECIFIC RANGE OF POINTS
C IN THE SPECTRAL RANGE.  THE RANGE IS ENTERED AS A NUMBER BETWEEN THE
C LIMITS OF 0 TO 3.14.  THAT'S PI.
C
C THE NEW STEP IS FOUND, AND THE LOWER IS THE STARTING POINT TO EVALUATE
C THE MAGNITUDE AND PHASE, ERROR ETC.
C

SUBROUTINE SETRANGE(STEP,NPOINTS,RANGE,LOWER)

REAL STEP,UPPER,LOWER,RANGE,PI,RANGE
INTEGER NPOINTS
BYTE ANS

PI = 3.1415926537

  IF ( NPOINTS .LE. 0 ) THEN

WRITE(6,1)'MUST HAVE A VALUE FOR THE NUMBER OF POINTS:'
RETURN

  ENDIF


41 WRITE(6,1)'DO YOU WANT TO USE 0.0 - 3.14; NYQUIST RANGE:    [RET]:'
WRITE(6,1)'DO YOU WANT TO USE A NARROW BAND IN NYQUIST RANGE: [Y]:'
```

```
READ(5,2,END=99,ERR=41)ANS

IF ( ANS .EQ. ' ' ) THEN
STEP = PI/NPOINTS
RANGE = PI
LOWER = 0.0
END IF


IF ( ANS .EQ. 'y' )THEN
ANS = 'Y'
END IF


IF ( ANS .EQ. 'Y' ) THEN

42 WRITE(6,1)'THE RANGE MUST BE WITHIN ZERO - PI: (0. TO 3.14159265)'
WRITE(6,1)'ENTER THE LOWER BOUND WITH DECIMAL :'
READ(5,5,END=99,ERR=42)LOWER


IF ( (LOWER .GT. (0.0)) .AND. (LOWER .LE. PI) ) THEN
CONTINUE
ELSE
WRITE(6,1)'BE CAREFUL.  TRY AGAIN. AND READ THE MESSAGES.....'
WRITE(6,1)'                          HIT RETURN KEY TO CONTINUE...:'
READ(5,2,END=99,ERR=41)ANS
LOWER = 0.0
UPPER = PI
GOTO 45
END IF


WRITE(6,6)LOWER

WRITE(6,1)'ENTER THE UPPER BOUND WITH DECIMAL :'
READ(5,5,END=99,ERR=42)UPPER


IF ( (UPPER .GT. (0.0)) .AND. (UPPER .LE. PI)
     &                          .AND. (UPPER .GT. LOWER) )THEN

CONTINUE

ELSE
```

```
WRITE(6,1)'BE CAREFUL.  TRY AGAIN. AND READ THE MESSAGES.....'
WRITE(6,1)'                          HIT RETURN KEY TO CONTINUE...:'
READ(5,2,END=99,ERR=41)ANS

LOWER = 0.0
UPPER = PI

END IF


WRITE(6,7)UPPER

WRITE(6,4)
45 RANGE = UPPER - LOWER
STEP = RANGE / NPOINTS


ENDIF


WRITE(6,4)


1 FORMAT(/,A,$)
2 FORMAT(A)
3 FORMAT('THE NUMBER OF SPECTRAL POINTS IS SET AT "',I4,'"',/)
4 FORMAT(/,/)
5 FORMAT(F15.9)
6 FORMAT('THE LOWER BOUND IS GOING TO BE =',F12.8)
7 FORMAT('THE UPPER BOUND IS GOING TO BE =',F12.8)

98 RETURN
99 END


C ***************************************************************************
C ***************************************************************************
C THIS IS CALLED FROM THE MAIN ROUTINE.  IT DOES A DIRECT FORM CALCULATION
C ON THE POLES.  THE DIRECT FORM FILTER IS MUCH SIMPLIER THAN CASCADE.
C DEN IS THE VALUE IN THE DENOMENATOR FOR THE SPECIFIC SPECTRAL POINT
C THAT IS EVALUATED.  THE SPECTRAL POINT IS W (BETWEEN 0 - PI).
C
SUBROUTINE DIRECTDEN(W,NY,A,DEN)

COMPLEX DEN
```

```
REAL A(*),W

   DEN = (0.,0.)
   DO JJ = 1,NY
    DEN = DEN + A(JJ) * CMPLX(COS(W*(JJ-1)),-SIN(W*(JJ-1)))
   END DO

RETURN
END
C---
C ***************************************************************************
C THIS IS CALLED FROM THE MAIN ROUTINE.  IT DOES A DIRECT FORM CALCULATION
C ON THE ZEROS.  THE DIRECT FORM FILTER IS MUCH SIMPLIER THAN CASCADE.
C NUM IS THE VALUE IN THE NUMERATOR FOR THE SPECIFIC SPECTRAL POINT
C THAT IS EVALUATED.  THE SPECTRAL POINT IS W (BETWEEN 0 - PI).
C

SUBROUTINE DIRECTNUM(W,NX,B,NUM)

COMPLEX NUM
REAL B(*),W

   NUM = (0.,0.)
   DO JJ = 1,NX
   NUM = NUM + B(JJ) * CMPLX(COS(W*(JJ-1)),-SIN(W*(JJ-1)))
   END DO

RETURN
END
C---
C ***************************************************************************
C THIS IS CALLED BY THE MAIN ROUTINE.  IT DOES CASCADE FORM CALCULATIONS
C AS YOU CAN SEE IT IS MORE INVOLVED THAN THE DIRECT FORM.
C THE SPECIFIC SPECTRAL POINT IS EVALUATED FOR THE CASCADE FILTER.
C THE SPECTRAL POINT IS W AND IS BETWEEN ( 0 - PI ).  THE VALUSE IS
C RETURNED IN NUM.  THIS ROUTINE TAKES THE COEFFICIENT ARRAY GIVEN TO IT.
C THE COEFFICIENT ARRAY IS ONE OF TWO ARRAYS: THE UNQUANTIZED ARRAY B(*)
C OR THE QUANTIZED ARRAY THAT'S PASSED TO THIS SUBROUTINE AS BQ(*).
C
SUBROUTINE CASCADENUM(W,NX,B,NUM,NUMBITS)

COMPLEX NUM,NCAS,T1,T2,T3
REAL B(*),W
INTEGER NX,SECTIONS,ZERO,ONE,TWO,NUMBITS
```

```
ZERO = 0
ONE = 1
TWO = 2
SECTIONS = NX/3
NCAS = (1.0,1.0)

J = 1
N1 = (J * 3) - 2
N2 = (J * 3) - 1
N3 = (J * 3) - 0
T1 = B(N1) * CMPLX(COS(W * ZERO),-SIN(W * ZERO))
T2 = B(N2) * CMPLX(COS(W * ONE),-SIN(W * ONE))
T3 = B(N3) * CMPLX(COS(W * TWO),-SIN(W * TWO))
NCAS = (T1 + T2 + T3)


IF ( SECTIONS .GE. 2 ) THEN

DO J = 2, SECTIONS

N1 = (J * 3) - 2
N2 = (J * 3) - 1
N3 = (J * 3) - 0
T1 = B(N1) * CMPLX(COS(W * ZERO),-SIN(W * ZERO))
T2 = B(N2) * CMPLX(COS(W * ONE),-SIN(W * ONE))
T3 = B(N3) * CMPLX(COS(W * TWO),-SIN(W * TWO))
NCAS = NCAS * (T1 + T2 + T3)

END DO


ENDIF


NUM = NCAS

1 FORMAT(/,A,$)
2 FORMAT(/,'THE NUM IS ',F15.5,F15.5)
RETURN
END
C---
C ***************************************************************************
C THIS IS CALLED BY THE MAIN ROUTINE.  IT DOES CASCADE FORM CALCULATIONS
C AS YOU CAN SEE IT IS MORE INVOLVED THAN THE DIRECT FORM.
```

```
C THE SPECIFIC SPECTRAL POINT IS EVALUATED FOR THE CASCADE FILTER.
C THE SPECTRAL POINT IS W AND IS BETWEEN ( 0 - PI ).  THE VALUSE IS
C RETURNED IN DEN.  THIS ROUTINE TAKES THE COEFFICIENT ARRAY GIVEN TO IT.
C THE COEFFICIENT ARRAY IS ONE OF TWO ARRAYS: THE UNQUANTIZED ARRAY A(*)
C OR THE QUANTIZED ARRAY THAT'S PASSED TO THIS SUBROUTINE AS AQ(*).
C

SUBROUTINE CASCADEDEN(W,NY,A,DEN,NUMBITS)

COMPLEX DEN,DCAS,T1,T2,T3
REAL A(*),W
INTEGER NY,SECTIONS,ZERO,ONE,TWO,NUMBITS


SECTIONS = NY/3
ZERO = 0
ONE = 1
TWO = 2
DCAS = (1.0,1.0)
J = 1
N1 = (J * 3) - 2
N2 = (J * 3) - 1
N3 = (J * 3) - 0
T1 = A(N1) * CMPLX(COS(W * ZERO),-SIN(W * ZERO))
T2 = A(N2) * CMPLX(COS(W * ONE),-SIN(W * ONE))
T3 = A(N3) * CMPLX(COS(W * TWO),-SIN(W * TWO))
DCAS = (T1 - T2 - T3)


IF ( SECTIONS .GE. 2 ) THEN

DO J = 2, SECTIONS

N1 = (J * 3) - 2
N2 = (J * 3) - 1
N3 = (J * 3) - 0
T1 = A(N1) * CMPLX(COS(W * ZERO),-SIN(W * ZERO))
T2 = A(N2) * CMPLX(COS(W * ONE),-SIN(W * ONE))
T3 = A(N3) * CMPLX(COS(W * TWO),-SIN(W * TWO))
DCAS = DCAS * (T1 - T2 - T3)

END DO


ENDIF
```

```
DEN = DCAS
c WRITE(6,2)DEN
1 FORMAT(/,A,$)
2 FORMAT(/,'THE DEN IS ',F15.5,f15.5)
3 FORMAT('THE SECTIONS IS ',I)

RETURN
END

C---
C ***************************************************************************
C IT'S MUCH EASIER TO HAVE COEFFICIENTS SAVED IN A FILE.  THEN YOU CAN
C ALWAYS HAVE THEM, AND BE ABLE TO MESS WITH THEM.
C
SUBROUTINE SAVECOEF(FNAME,NX,NY,A,B)

CHARACTER*15 FNAME
INTEGER NX, NY
REAL B(512),A(512)
BYTE ANS

52 WRITE(6,4)
41 WRITE(6,1) '  ***  THIS ROUTINE WILL STORE YOUR COEFFICIENTS   ***'
WRITE(6,1) '  ***  TO THE FILE SHOWN OR YOU MAY ENTER A NEW    ***'
WRITE(6,1) '  ***  FILE NAME TO STORE YOUR COEFFICIENTS IN.    ***'
WRITE(6,6) 'THE NAME OF THE FILE TO STORE YOUR COEFFICIENTS IS :',FNAME
WRITE(6,1) '------ IS THIS A CORRECT FILENAME TO USE ----- '
WRITE(6,4)
WRITE(6,1) 'THIS FILENAME IS CORRECT; PROCEED TO STORE: [RETURN KEY]:'
WRITE(6,1) 'THIS IS NOT RIGHT, I WANT TO ENTER NEW FILE:[(W)rong]:'

READ(5,5,END=99,ERR=41)ANS


IF ( (ANS .EQ. 'w') .OR. (ANS .EQ. 'W') ) THEN

WRITE(6,1)'ENTER THE NEW FILE NAME:'
READ(5,5,END=99,ERR=41)FNAME
WRITE(6,6)'THE FILE NAME IS CALLED:',FNAME
WRITE(6,1)'SHALL I PROCEED TO SAVE COEFFICIENTS? [Y]:'
WRITE(6,1)'SHALL I START THIS ROUTINE AGAIN? [RETURN KEY]:'
READ(5,5,END=99,ERR=41)ANS
```

```
  IF ( (ANS .EQ. 'Y') .OR. (ANS .EQ. 'y') ) THEN
CONTINUE
  ELSE
GOTO 41
  ENDIF

ELSE

CONTINUE

ENDIF


OPEN(10,NAME=FNAME,STATUS='UNKNOWN',ERR=52)
WRITE(10,2)NX
WRITE(10,2)NY

DO I = 1,NX
WRITE(10,3)B(I)
END DO

DO I = 1,NY
WRITE(10,3)A(I)
END DO

CLOSE(10)

WRITE(6,1) 'SAVING COEFFICIENTS COMPLETE; COEFFICIENTS WRITTEN.'
WRITE(6,6) 'THE NAME OF YOUR COEFFICIENT FILE IS :',FNAME
WRITE(6,4)
WRITE(6,4)
C--- END OF INPUT ROUTINE, THE COEFFICIENTS HAVE BEEN READ IN.
1 FORMAT(/,A,$)
2 FORMAT(I3)
3 FORMAT(F14.7)
4 FORMAT(/,/)
5 FORMAT(A,$)
6 FORMAT(/,A,A,/)
20 FORMAT(2X,'A(',I3,') = ',F14.8)
21 FORMAT(2X,'B(',I3,') = ',F14.8)

99 RETURN
END
```

```
C---
C ***********************************************************************
C THIS ROUTINE WILL MAKE SURE THE COEFFICIENTS ARE NEVER GREATER THAN'
C THE VALUE ONE.  TO IMPLEMENT COEFFICIENTS IN A FILTER, USING TWO'S
C COMPLEMENT, THE COEFFICIENTS ARE ALL LESS THAN ONE.  THIS CREATES THE
C NORMALIZED COEFFICIENT FILE.  IT ALSO CAN INCREASE THE SIZE OF THE
C COEFFICIENTS IF THEY ARE ALL LESS THAN ONE, SO AT LEAST ONE COEFFICIENT
C WILL BE THE VALUE OF ONE.  THIS MAXIMIZES THE DYNAMIC RANGE OF THE
C NUMBERS USED IN THE COEFFICIENT FILE.
C
C THE USER WILL BE ABLE TO SAVE THE MODIFIED COEFFICIENTS INTO A NEW FILE.
C THIS WAY, A USER CAN HAVE A NON-NORMALIZED COEFFICIENT FILE.
C AND A USER CAN HAVE A NORMALIZED COEFFICIENT FILE.
C
C NOTE::::::
C THE USE OF TWO'S COMPLEMENT ABSOLUTELY REQUIRES NUMBERS LESS
C THAN THE VALUE OF ONE.............
C
  SUBROUTINE NORMALIZATION(FNAME,NX,NY,A,B)

BYTE ANS
CHARACTER*15 FNAME,FNAMET
INTEGER NX, NY, IHIGH
REAL B(512),A(512),BIGB,BIGA,TOP

53 IHIGH = 1
WRITE(6,4)
WRITE(6,1) 'THIS ROUTINE WILL NORMALIZE THE FILTER COEFFICIENT FILE.'
WRITE(6,1) '     --------------  IF NECESSARY  --------------'
WRITE(6,1)'*******  COEFFICIENTS MUST INCLUDE DECIMAL POINTS  *******'

WRITE(6,1)'THE INPUT FILE WITH THE COEFFICIENTS WILL BE READ IN.'
WRITE(6,7)
WRITE(6,6) 'THE NAME OF THE FILE WITH THE COEFFICIENTS IS: ',FNAME

WRITE(6,7)

   OPEN(4,NAME=FNAME,STATUS='OLD',ERR=53)

   READ(4,2,END=99,ERR=53)NX
   READ(4,2,END=99,ERR=53)NY

DO J = 1,NX
READ(4,3,END=99,ERR=53)B(J)
```

```
WRITE(6,21)J-1,B(J)

 IF ( B(J) .GT. 1.0 ) THEN
IHIGH = 2
 END IF

END DO


DO J = 1,NY
READ(4,3,END=99,ERR=53)A(J)
WRITE(6,20)J-1,A(J)

 IF ( A(J) .GT. 1.0 ) THEN
IHIGH = 2
END IF

END DO

    CLOSE(4)

 IF ( IHIGH .EQ. 1 ) THEN

42 WRITE(6,1)'NORMALIZATION ROUTINE NOT NEEDED; COEFFICIENTS
     $                                    BETWEEN -1.0 AND 1.0 .'
WRITE(6,1)'........YOUR COEFFICIENT FILE IS NOT OUT-OF-RANGE .........'
WRITE(6,1)'------------------------------------------------------------'
WRITE(6,4)
WRITE(6,1)'DO YOU WANT TO ADJUST COEFFICIENTS TO MAXIMIZE THE RANGE?'
WRITE(6,1)'COEFFICIENTS WILL HAVE A MAXIMUM VALUE OF "1.0"           '
WRITE(6,1)'----- MAXIMUM VALUE OF "1.0" DESIRED ----: ENTER     [Y]'
WRITE(6,1)'----- LEAVE COEFFICIENTS ALONE ------     : ENTER[ANYKEY]'
READ(5,8,END=99,ERR=42)ANS

IF ( (ANS .EQ. 'Y') .OR. (ANS .EQ. 'y') ) THEN

CONTINUE

ELSE

WRITE(6,4)
WRITE(6,4)
RETURN

END IF
```

```
END IF



43 WRITE(6,1)'  THE  -TARGET-  FILE IS READ IN NEXT    ........'

WRITE(6,1) 'ENTER NAME OF TARGET FILE FOR NORMALIZED COEFFICIENTS...:'
 READ(5,5,END=99,ERR=43) FNAMET
WRITE(6,7)

WRITE(6,6) 'TARGET FILE FOR THE COEFFICIENTS IS: ',FNAMET
WRITE(6,5) '--------   SHALL I PROCEED   --------        [Y] YES...:'
WRITE(6,8) '                                  [ANY OTHER KEY] NO ...:'

READ(5,8,END=99,ERR=53) ANS

 IF ( (ANS .EQ. 'Y') .OR. (ANS .EQ. 'y') ) THEN


C*************************************************************************

IF ( FNAMET .EQ. ' ' ) THEN

FNAMET = 'filter?'
RETURN

ENDIF

C *************************************************************************

BIGB = ABS(B(1))
BIGA = ABS(A(1))

DO I = 1,(NX-1)
  IF ( ABS(B(I+1)) .GE. ABS(BIGB) ) THEN
BIGB = B(I+1)
ENDIF
END DO


DO I = 1,(NY-1)
  IF ( ABS(A(I+1)) .GE. ABS(BIGA) ) THEN
BIGA = A(I+1)
ENDIF
```

```
END DO


BIGB = ABS(BIGB)
BIGA = ABS(BIGA)


IF ( BIGA .GE. BIGB ) THEN

TOP = BIGA

ELSE

TOP = BIGB

END IF


DO I = 1,NX

B(I) = B(I)/TOP

END DO


DO I = 1,NY

A(I) = A(I)/TOP

END DO

OPEN(10,NAME=FNAMET,STATUS='UNKNOWN',ERR=98)

WRITE(10,2)NX
WRITE(10,2)NY

DO I = 1,NX
WRITE(10,3)B(I)
END DO

DO I = 1,NY
WRITE(10,3)A(I)
END DO
```

```
CLOSE(10)

WRITE(6,1) 'NORMALIZATION ROUTINE NOW COMPLETE; TARGET FILE WRITTEN.'
WRITE(6,7)
WRITE(6,6) 'THE NAME OF YOUR NORMALIZED COEFFICIENT FILE IS :',FNAMET
        WRITE(6,1)'*** MENU OPTION #5 UPDATED WITH TARGET COEFFICIENT FILE.***'
WRITE(6,1)  '                 .............[ANY KEY TO CONTINUE]'
FNAME = FNAMET
READ(5,5,END=99,ERR=98)ANS
WRITE(6,4)
WRITE(6,4)

ELSE

RETURN

ENDIF


C--- END OF INPUT ROUTINE, THE COEFFICIENTS HAVE BEEN READ IN.
1 FORMAT(/,A,$)
2 FORMAT(I3)
3 FORMAT(F15.10)
4 FORMAT(/,/)
5 FORMAT(A)
6 FORMAT(A,A,/)
7 FORMAT(/)
8 FORMAT(A,$)
9 FORMAT(A,I)
20 FORMAT(2X,'A(',I3,') = ',F14.8)
21 FORMAT(2X,'B(',I3,') = ',F14.8)

98 RETURN
99 END

C ***************************************************************************
C ***************************************************************************
C AM I TO COMMENT ON THE COMMENTS SECTION?
C WELL I GUESS I SHOUD.  THIS IS A NICE COMMENT SECTION THAT WILL PRINT
C OUT TO THE SCREEN FOR A USER TO GET A PICTURE OF THIS PROGRAM'S I/O
C REQUIREMENTS.  I'VE WRITTEN ERROR CHECK ROUTINES, SO IF A USER TRIES TO
C VIOLATE SOMETHING, THIS SOFTWARE WILL CATCH IT TO FORCE PROPER USEAGE.
C
SUBROUTINE HELPONINPUT
```

```
WRITE(6,1)' ________FOR A  CASCADE FILTER___________'
WRITE(6,1)'BE SURE TO ENTER AN EQUAL NUMBER OF NUMERATOR AND '
WRITE(6,1)'DENOMINATOR SECTIONS FOR YOUR FILTER.'
WRITE(6,1) ' '
WRITE(6,1)'  BE SURE TO INPUT COEFFICIENTS LIKE IN EQ. 6.23 '
WRITE(6,1)'   FROM OPPENHEIM AND SCHAFER. '
WRITE(6,1)'  BUT INCLUDE THE '1.0' IN THE DENOMINATOR AS A COEFFICIENT'
WRITE(6,1)'  EACH SECTION WILL HAVE SIX COEFFICIENTS.................'
WRITE(6,1)'         ************** EXAMPLE ***************  '
WRITE(6,1)'             EACH SECTION IS LOOKS LIKE THIS    '
WRITE(6,1)' '
WRITE(6,1)'          (B_0 + B_1*Z^-1 + B_2*Z^-2)   (...)    '
WRITE(6,1)' H(Z) =   ------------------------- * ----- * ...'
WRITE(6,1)'          (A_0 - A_1*Z^-1 - A_2*Z^-2)   (...) '
WRITE(6,1)' '
WRITE(6,1)'                              note: position A_0 = 1.0 always'
WRITE(6,1)'                                                '
WRITE(6,4)
WRITE(6,1)'_____________FOR A DIRECT FORM FILTER_____________'
WRITE(6,1)' THE X COEFFICIENTS ARE ON THE RIGHT SIDE OF DIFFERENCE EQUATION'
WRITE(6,1)'  THE Y COEFFICIENTS ARE ON THE LEFT SIDE OF DIFFERENCE EQUATION'
WRITE(6,1)'  BE SURE TO INPUT COEFFICIENTS LIKE IN EQ. 5.16 '
WRITE(6,1)'   FROM OPPENHEIM AND SCHAFER.................. '
WRITE(6,1)'   '
WRITE(6,1)'         ************** EXAMPLE ***************  '
WRITE(6,1)'             EACH FILTER LOOKS LIKE THIS    '
WRITE(6,1)' '
WRITE(6,1)'    Y(Z)*A_0 + Y(Z^-1)*A_1 + Y(Z^-2)*A_2 + ...   =   '
WRITE(6,1)'                                      '
WRITE(6,1)'    X(Z)*B_0 + X(Z^-1)*B_1 + X(Z^-2)*B_2 + ...       '
WRITE(6,1)' '
WRITE(6,1)'                                   '
WRITE(6,1)'                                   HIT RETURN TO CONTINUE...:'
READ(5,5)

WRITE(6,4)
WRITE(6,1)'_____________INPUT FILE FOR CASCADE COEFFICIENTS__________'
WRITE(6,1)'   A SAMPLE FOURTH ORDER FILTER FILE IS THE FOLLOWING:  '
WRITE(6,1)'       ************** EXAMPLE  FILE  ***************  '
WRITE(6,1)'  '
WRITE(6,1)'6 note:number of "X" coefficients'
WRITE(6,1)'6 note:number of "Y" coefficients'
WRITE(6,1)'0.135843 note:b_0'
```

```
WRITE(6,1)'0.026265 note:b_1'
WRITE(6,1)'0.135843 note:b_2'
WRITE(6,1)'0.278901 note:b_3'
WRITE(6,1)'-0.444500 note:b_4'
WRITE(6,1)'0.278901 note:b_5'
WRITE(6,1)'1.0 note:a_0'
WRITE(6,1)'0.738409 note:a_1'
WRITE(6,1)'-0.850835 note:a_2'
WRITE(6,1)'1.0 note:a_3'
WRITE(6,1)'0.960374 note:a_4'
WRITE(6,1)'-0.860000 note:a_5'
write(6,4)
WRITE(6,1)'_____________INPUT FILE FOR DIRECT COEFFICIENTS_____FIR____'
WRITE(6,1)'   A SAMPLE DIRECT FORM FIR FILTER FILE IS THE FOLLOWING:  '
WRITE(6,1)'       ************** EXAMPLE FILE ***************  '
WRITE(6,1)'  '
WRITE(6,1)'5 note:number of "X" coefficients'
WRITE(6,1)'1 note:number of "X" coefficients'
WRITE(6,1)'1.359657E-3 note:b_0'
WRITE(6,1)'-1.616993E-3 note:b_1'
WRITE(6,1)'-7.738032E-3 note:b_2'
WRITE(6,1)'-2.686841E-3 note:b_3'
WRITE(6,1)'1.255246E-2 note:b_4'
WRITE(6,1)'1.0 note:a_0'
WRITE(6,4)
WRITE(6,1)'                                    HIT RETURN TO CONTINUE...:'
READ(5,5)
WRITE(6,4)

WRITE(6,1)'_____________INPUT FILE FOR DIRECT COEFFICIENTS____IIR_____'
WRITE(6,1)'   A SAMPLE DIRECT FORM IIR FILTER FILE IS THE FOLLOWING:  '
WRITE(6,1)'       ************** EXAMPLE FILE ***************  '
WRITE(6,1)'  '
WRITE(6,1)'5 note:number of "X" coefficients'
WRITE(6,1)'3 note:number of "X" coefficients'
WRITE(6,1)'1.359657E-3 note:b_0'
WRITE(6,1)'-1.616993E-3 note:b_1'
WRITE(6,1)'-7.738032E-3 note:b_2'
WRITE(6,1)'-2.686841E-3 note:b_3'
WRITE(6,1)'1.255246E-2 note:b_4'
WRITE(6,1)'0.265234 note:a_0'
WRITE(6,1)'-0.850835 note:a_1'
WRITE(6,1)'0.0262341 note:a_2'
WRITE(6,4)
WRITE(6,1)'                                    HIT RETURN TO CONTINUE...:'
```

```
      READ(5,5)

      WRITE(6,1)'                                                    '
      WRITE(6,1)'  '
      WRITE(6,1)'  '


    1 FORMAT(/,A,$)
    4 FORMAT(/,/)
    5 FORMAT(A)


      RETURN
      END
C     ------------------------------------------------------------------
C THANK GOD THAT I'M DONE. THIS IS A REAL NICE PROGRAM TOO.
```

Appendix B. *User's Manual To Use The Digital Software Analyzer Tool*

The Digital Software Analyzer Tool performs two functions. The first is the computation of the filter's response when the coefficients are represented by a limited number of bits. The second computation finds the roundoff power in the generated by the structure of the digital filter.

The user of the Digital Software Analyzer Tool will find the program to be some what self teaching. The program when run will display a main menu. The main menu will continue to redisplay after the user makes the selections desired. Each of the options for the tool are clear in their operation. The user will need to read the main menu options to get an idea of the capabilities of the analyzer tool. The main menu also has a selection to choose from that will provide a set of help screens. These help screens will show the user by example the structure of the filter's coefficients that the program will expect in terms of position and sign.

B.1 The Input Needed And The Output Generated

The input to the digital filter analysis tool is a few numbers that represent the filter coefficients. These coefficients can be input using up to seven significant digits. That allows 24-bits to represent the accuracy of the number with a very large range (single precision). The output from the digital filter analysis tool are simply data files. These output files contain the results of a filter simulation. Three basic types of files are produced. The magnitude file shows the response of the filter to different frequencies. The phase file shows the phase relationship to frequencies. This is a way to verify linear phase characteristics. The error file shows the user the extent of difference between the best accuracy available from the input coefficients to the accuracy available from the number of "what if I had so many bits available" to use in the filter implementation. The error file is a linear difference between the unquantized and the quantized versions of the magnitude responses.

The output file that has the magnitude response is listed as two types. The two types correspond to the unquantized and the quantized versions. When the digital filter analysis tool is used, the opening menu displays all the options for the user to select from. Two options allow the user to associate a two digit number with the output files. One of these two options appends a two digit number with the unquantized simulation results and the other option does the same for the quantized simulation results. The resulting file structure is quite clean. A user can immediately tell if a file is the unquantized or the quantized simulation results.

B.2 How The Digital Filter Analysis Tool is Organized

This software is organized into three major blocks. The first block is the opening section where the screen is written to with the options for the user to choose from. The first block also includes the controlling section. The controlling section calls the appropriate subroutines specified by the option selection. The second major block is the main section of the program. The main section of the program is where the simulation is actually performed. The main section does the computation of the magnitude, phase, and error. The last major block of the program is where all the subroutines are found. With a program of this size, subroutines were used to keep the tasks in a smaller, understandable section of code. There could be more subroutines to split out the tasks even further.

Throughout the program are error catching routines. Error catching routines check for all input and output interfacing. These will include limits, type, and physical correctness in the structure of the input keystrokes. Other error checks look for other more subtle areas that may or may not cause immediate problems. One such area is the requirement for the same number of cascaded second-order sections.

B.2.1 The Controlling Section This major block is called the controlling section. In the ASCII code written in fortran, a long section of comments explain the interface requirements. The comments also include the general format used for the coefficients. Data

files can be used with the digital filter analysis tool and are recommended. The use of files to keep the coefficients for the filter is a handy way to keep many filters available. The comments also include the general format used for the coefficients.

There are two basic digital filter structures. The first is the Direct structure and the second is the Cascade structure. It's important to understand the format of these two structures. Not only are the format of these two structures found in the the Major Block, but I also have them in two other areas of the program. Each of the areas send to the screen both information about the structure and the transfer function or difference equation.

When a user selects the type of filter structure to use, the program writes the format of that particular structure to the screen. The program also sends information about the format, an example of the format, and where to look for further information. Since the format of the input files is so important, their structure is shown to the user on the screen device.

B.2.2 Direct Structure Required to Build Coefficient Files There are many ways to organize a file that contains the coefficients for a direct form. Only one form is allowed. The different forms can be in terms of what domain the system is to be characterized in or the forms can be involved with the sign of the coefficients to the terms in the equation.

The difference equation as used by this digital filter analysis tool is written down with the coefficients for the "y" terms on the left side of the difference equation and the coefficients for the "x" terms on the right side of the difference equation. The following difference equation shows the form as

$$a_0 y[n] + a_1 y[n-1] + a_2 y[n-2] + \cdots = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] + \cdots .$$

The data file containing these coefficients must have their coefficients with the appropriate sign shown by this equation. The data file is organized in a straight forward manner.

The data file will always have four parts to it.

1. The first number in a data file is the number of X coefficients.

2. The second number in a data file is the number of Y coefficients.

3. The next set of numbers are the X coefficients.

4. The last set of numbers are the Y coefficients.

The following example shows the structure of the direct form. This example has five "x" coefficients and three "y" coefficients.

```
5
3
0.235
0.412
-0.15
0.023
0.623
1.000
0.375
-0.264
```

The difference equation written using the coefficients shown above would take on this form:

$$1.0y[n] + 0.375y[n-1] + -0.264y[n-2] =$$
$$0.235x[n] + 0.412x[n-1] + -0.15x[n-2] + 0.023x[n-3] + 0.623x[n-4].$$

The coefficient for the $y[n]$ term is shown to be the value 1.0. This term is a gain factor for the filter. The software does have utilities to change this coefficient. The user will know that a normalization procedure has taken place. It's up to the user to track the gain. The reason for this is that the gain is placed after the filtering operation. The gain factor will not affect the performance of the filter except for saturation problems (that's a different issue). As a general rule of thumb, filters are designed with this term equal to one and

the other coefficients are built from this normalization basis. This issue becomes important when entering the coefficients for the cascade designed filter.

B.2.3 Cascade Structure Required to Build Coefficient Files The Cascade sections consist of second order sections. Each section in the numerator and each section in the denominator must be a second order polynomial. The number of ways to organize a data file for the coefficients pertaining to the cascade sections are varied. Since the form of a cascade section is built into fractions, one part of the difference equation must be divided by the other part of the difference equation. The form of the direct difference equation is used to build the cascade sections. The Y terms are first subtracted from the left side of the difference equation. Then both the numerator and the denominator are divided by the Y terms. The result is a form that can be a standard for input.

The form of a second order cascade section is:

$$H(z) = \frac{b_1 + b_2 z^{-1} + b_3 z^{-2}}{1 - a_2 z^{-1} - a_3 z^{-2}}.$$

This is only a single cascade section. Additional cascade sections will be multiplied to the other cascade sections. The coefficients for the X terms are associated with the"b" coefficient terms. The coefficients for the Y terms are associated with the "a" coefficient terms. Cascading two second order sections results in

$$H(z) = \left(\frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}}\right)\left(\frac{b_3 + b_4 z^{-1} + b_5 z^{-2}}{1 - a_4 z^{-1} - a_5 z^{-2}}\right).$$

If there are an odd number of poles or zeros in the filter design, the coefficients can be zero to maintain this structure. The program *requires that there be an equal number of numerator sections and denominator sections.* Setting the coefficients to the additional sections needed to be a value of zero, the requirement of equal number of cascade sections will be met. This requirement stems from the subroutines used in the program.

Another requirement is the necessity of three coefficients for each part of the cascade section. *The numerator must have three coefficients associated with it, and the denominator must have three coefficients associated with it as will.* Also the user must notice the sign in the denominator for the cascade section. If the design yields these sign conventions then no adjustment is necessary. However, if the design implements the cascade section with all plus signs, then the coefficients must be of different sign when input to a data file.

The following is an example of a cascade designed filter. The cascade format is well suited for implementation using the canonic structure. This filter consists of two sections. The first number tells the number of the X terms. The second number tells the number of the Y terms. The next X numbers are the coefficients for the numerator sections, and the next Y numbers are the coefficients for the denominator sections.

```
6
6
 0.1358430
 0.0262650
 0.1358430
 0.2789010
-0.4445000
 0.2789010
 1.0000000
 0.7384090
-0.8508350
 1.0000000
 0.9603740
-0.8600000
```

It should be some what obvious of the value of the $y[n]$ coefficient terms in each section, since they are all equal to one. The format of the data files for this cascade form is exactly like the format for the direct form. These files can be written with any ASCII editor. The digital filter analysis tool will read these files in or it can write the coefficient data to a file.

B.3 Controlling Section, Main Menu Options To Run The Digital Analyzer

The user will be presented with a menu screen. The options are available to help conform the type of analysis to be exactly what the user desires.

Some variables are not computed within the program when first used. These variables are initialized to values needed to start the simulation with appropriate default variables. A do while loop surrounds the Major Block of code to control the flow of the program. The do while loop is the means to keep the programs revolving around the main menu for the user to pick the item of choice.

The main menu is the means for the user of the digital analyzer tool to make the sections appropriate for the simulation desired. The following is a picture of the screen that shows how the main menu is constructed. The main menu has the present values of the choices shown. Often, there exits a sub-menu below the main menu choice options for the user to continue to make desired selections adding flexibility to the simulation process. What is shown is the default menu. The choices presented under each main menu choice are explained in the next sections.

B.3.0.1 Enter The Type Of Filter, Or The Conversion Process; Main Menu Option 1. The user can select from two types of filter structures. The first structure is the cascade structure. The second structure is the direct structure. Both of these options have an example of the format printed to the screen.

There is a sub-option menu that performs a conversion process on a filter structure. The conversion process takes a cascade structure and converts it into the direct structure. This conversion process is important to see the effects of implementation with both types of structures. The conversion process is explained with details in Section C.1.1.1. The conversion process takes the second order cascade sections and multiplies them together to form one higher order polynomial. The type of filter is updated, the coefficient file is re-written, and control returns to main menu.

```
*DISCRETE SIGNAL PROCESSING, FILTER DESIGN, Capt P. Choate*

(1) ENTER TYPE OF FILTER, OR CONVERSION PROCESS :...
TYPE OF FILTER DESIRED IS ..... DIRECT FORM

(2) ENTER NUMBER OF BITS FOR COEFFICIENTS QUANTIZATION: ...
NUMBER BITS FOR COEFFICIENT QUANTIZATION IS..... 16

(3) ENTER NUMBER TO ASSOCIATE WITH THE OUTPUT FILES : ...
TWO DIGIT NUMBER TO TAG WITH OUTPUT FILES IS ..... 99

(4) ENTER NUMBER FOR UNQUANTIZED MAG/PHASE OUTPUT FILES : ...
TWO DIGIT NUMBER FOR UNQUANTIZED MAG/PHASE FILES IS.. 01

(5) ENTER COEFFICIENTS TO USE FROM FILE OR KEYBOARD : ...
COEFFICIENTS' FILE NAME IS..... ???????

(6) ENTER NUMBER OF SPECTRAL POINTS FOR CALCULATIONS : ...
NUMBER OF SPECTRAL POINTS FOR CALCULATIONS IS: ... 500

(7) ENTER THE NYQUIST BANDWIDTH TO VIEW FOR PLOTTING : ...
THE BANDWIDTH TO VIEW FOR PLOTS IS..... 3.1416 RAD/SEC

(8) CHECK COEFFICIENT FILE AND NORMALIZE IF NECESSARY: ...
WILL ALSO RENAME THE COEFFICIENT FILE.

(9) SAVE MY COEFFICIENTS TO A FILE IN THIS DIRECTORY :...
FOR KEYBOARD ENTERED COEFFICIENTS.

(10) FIND THE ROUND-OFF POWER ERROR FOR CASCADE .... :...
AND DIRECT DESIGNS; SWAP CASCADE SECTIONS ....

(11) HELP ON HOW TO FORMAT INPUT COEFFICIENT FILES.

(55) ——— ALL DONE ———

**** PLEASE ENTER NUMBER OR [RET] TO RUN PROGRAM *****
ENTER NUMBER [#]:
```

Figure B.1. Main Menu Options Screen for the Digital Filter Simulator

The conversion process is where the user can perform research on the implementation structure of a digital filter. This conversion utility will take the coefficients from the cascade structure and build a filter that uses the direct structure. The conversion process re-builds a new set of coefficients. The new coefficients computed during the conversion will not be saved to disk. However, the user will be prompted to save the newly created coefficients to the disk system under a new data file name.

B.3.0.2 Enter Number of Bits For Coefficient Quantization; Main Menu Option 2. The user can select any number of bits to use in the representation of the coefficients to use in the multipliers. Of course the number of bits is limited to the format used by the computer system. The format used by the software is single precision. Since it is IEEE format, the 24-bits used in the mantissa also include the sign bit. However, since the number is normalized in base two, the position to the left of the decimal is always assumed. Therefore a total of 24-bits are available for number representation. Typical values to try for a filter are 16-bits for high accuracy and down to four or five bits for the what if cases.

This option should be used with the option 3. A user can change the number of bits to use for the simulation and then also change the two digit number that tags on the output files. That way a user can have lots of output files different names corresponding to each of the simulation runs. For example, a user wants to run five simulations of 16-bits, 14-bits, 12-bits, 10-bits, and 8-bits. The user can simply use the main menu option 3 to tag with each run a different two digit number, like 01 through 05. This allows efficient analysis of the data.

B.3.0.3 Enter Number To Associate With The Output Files; Main Menu Option 3. The user can select any two digit number to associate with the output files. These output files all correspond to the quantized output files. They will include the magnitude response of the filter, the phase response of the filter, and the linear error measurement of the filter. These files will have a "q" attached as well as the two digit number. As a finer point, a user can input any ASCII character into this variable.

The use of this selection will help to keep the simulation runs organized. If a user wanted to run the simulation 50 times, the user could simply number the output files 01 through 50. See the note on the number to associate with unquantized magnitude and phase output files in Section B.3.0.4 since that is a different number and different output files.

B.3.0.4 Enter Number For Unquantized Mag/Phase Output Files; Main Menu Option 4. The user can select any two digit number to associate with the full single precision (unquantized) output files. There are two unquantized output files: the magnitude file and the phase file. Each of these files will always be written out to the disk during simulation. Even though a user will change the number of bits to be used for each simulation, these two output files will not change in value. The reason is that they are built from the use of a constant number of bits to represent the data using single precision numbers. So, when a user is looking at one type of filter (the same set of coefficients used for the simulations) these two unquantized data files will always be the same. When a user changes the coefficients, then the output to these two files will change.

A user can select a new two digit number to use with each different set of coefficients for the filter simulation. This two digit number will tag on files that do not have a "q" associated with them.

B.3.0.5 Enter Coefficients To Use From File Or Keyboard: Main Menu Option 5. The user can select the method to enter coefficients for the filter simulation. Two means are available to allow a user to enter the coefficient numbers. The first is from the keyboard and the second is from the disk file system.

If the user desires to enter the coefficients from the keyboard, prompts will direct the user on what to enter. The user is forewarned that real numbers must include a decimal point. Coefficients are all real numbers. The user must follow the sign conventions given in this software tool in order to see the results expected. The user can select the help option to see what filter coefficients look like and hints on where in the difference equation the coefficients are located to avoid sign problems.

If the user desires to have the program read in the coefficients from a disk file, then the user will simply enter the file name. The normalization routine will correct all out of bound conditions.

For the benefit of the user, the coefficients will be displayed to the screen whether they are entered from the keyboard or read in from the file. The format that the coefficients take will directly depend on the type of filter structure chosen. Option one gives more information on this. If a user has the direct form chosen (also the default setting) then the coefficients will be displayed to the screen in a list format. However, if the type of filter chosen is the cascade filter, then the coefficients will be displayed in a form suited directly for that structure. In particular, each section will be displayed as a unique cascade unit and be labeled by section number.

B.3.0.6 Enter Number Of Spectral Points For Calculations; Main Menu Option 6. All of the output files must have a certain length. The length is the number of spectral points input to the program. The default length is 500 spectral points. The plots are done in a normalized fashion ranging from zero to the value π. The program will not compute the values from π to 2π since that range is an image of the first half of the spectral information.

To interpret the normalized frequency space, a user must have the value of the proposed sampling rate to be used with the digital filter. The sampling rate is the key for the frequency space, since π is half the sampling rate. A designer could change the sampling rate and effectively move about the magnitude plot to optimize the desired frequencies of interest.

The number of spectral points will determine the how many spectral points will be evaluated. Increasing the number of spectral points will show more detail in the magnitude response. However, the more spectral points desired will result in larger output files. The range can allow a user to focus in on certain areas of frequency to see the details in the filter's response.

B.3.0.7 Enter The Nyquist Bandwidth To View For Plotting; Main Menu Option 7. The digital filter analyzer produces output in a normalized frequency setting. That means the range of values for the plots are designed to be no larger that the range of π. This range covers the entire spectral range of possible spectral points for the simulation and the realization. By using the bandwidth adjustment, a user is able to focus on a region of interest in the magnitude response, the phase response, or the error plot.

The use of this option may include running the filter once to see the transition regions within the filter. Then, the user may focus the plots away from the stop regions and into the transition or pass band regions. The input to this option is a range between zero and π.

B.3.0.8 Check Coefficient File And Normalize If Necessary; Main Menu Option 8. This option will read in from a disk file the coefficient file named. As a warning to the users, the coefficients are real values in the software. Thus, they must include a decimal point. The use of two's complement to represent the numerics requires that they be of magnitude less than or equal to one. The routine will create a new data file for the coefficients when needed that will have a maximum value of one.

This routine will check the coefficients for the magnitude of the numbers. If the coefficients are a value greater than one, the normalization routine is initiated. If normalization is not needed, the option of normalizing the coefficients for maximum dynamic range is possible. This is needed especially when the coefficients from the design process are all less than $|1.0|$.

The user is presented with the option to save the coefficients to a data file with a new name whenever there is a change to the coefficient file. If the target file already exists with values that the user desires to keep and this routine is run, the write will destructively write over the 'old' values. Only in a few places does the program place restrictions on read/write operations. These will occur when the file is to exist and be read in. If the user incorrectly typed the name of the file, then the reading routine will catch this error and prompt the user.

B.3.0.9 Save My Coefficients To A File In This Directory; Main Menu Option 9. This routine will store the coefficients either into a file shown (default file name) or into a file named by the user. The write process is a destructive write process in that it will clear a previous file with the same file name.

This routine is useful when a user has some numbers for a filter and desires to use this program. The user can enter the coefficients to the program by using Main Menu Option 5. Then, in order to preclude entering the coefficients again, the coefficients can be saved to the disk. There's a limit of 15 characters for a file name.

B.3.0.10 Find The Round-off Power Error For Cascade And Direct Designs; Swap Cascade Sections; Main Menu Option 10. This routine will perform non-destructive manipulations to the coefficient file. When changes are made to the order of the coefficients, the original order will not be lost. Upon leaving from this routine, the original order of the coefficients is restored.

The roundoff power measurement can be found for both the direct and the cascade structure. The direct structure calculation can be done through the application of the theory (see Chapter III, roundoff power measurements). However, the calculations of the cascade roundoff power measurements are involved. This routine finds the roundoff measurement. The number produced is a measure of the goodness in the design. The user will see numbers in the range 10^{-8} watts. This routine finds the power figure by actually calculating the roundoff error in the filter.

The use of multiplications are the major cause of roundoff errors. The single precision number representation allows for 24-bits of accuracy. When another single precision number is multiplied by it, then the resultant will take 48-bits to hold. This case supposes the exponent is within bounds. The newly formed resultant must be stored back into single precision number representation. The process to accomplish this task of reducing the size of the resultant multiplication is called truncation and rounding. Effectively, the least sig-

nificant 24-bits are truncated to bring the result into a length compatible with the format. This rounding or truncating process is the cause of roundoff errors.

This routine will output to the screen a roundoff power measurement. The user can interpret the number as the amount of accuracy available for the filter design. So, if the roundoff power measurement is a number of magnitude 0.5 watts, the filter design would have at best only one bit of accuracy even though the length of the registers used are mucn higher.

The filter analysis used allows a user to see the effects of coefficient quantization separately from roundoff errors. Coefficient quantization is viewed and analyzed as a separate mechanism from roundoff errors. Likewise, the effects of roundoff error measurements are viewed and analyzed as a separate mechanism from the effects of coefficient quantization. By seeing the effects of each of these two problems, the designer will have a precise knowledge of the realized performance.

The user has the option to move cascade sections within the filter design. The concept is much like that of communications systems where the use of very small signal powers is required. The best performance of the entire system is to place the highest gain amplifier in the first position whether in the receiver. This process will produce the least amount of noise generated within the system design. The same is true in the process of the ordering of the cascade sections of the designed digital filter.

The design must find the best order for the filter for optimum performance. This will occur when the section with the higher gain factors are placed in the beginning of the filter. The lowest gain factor for the cascade filter is then to be placed at the end of the filter sections. The best process to use is to find the roots of the cascade sections and find those roots closest to the unit circle. The roots with poles and zeros closest to the unit circle will be correspond to the sections with the highest gain. These roots ought to be placed early in the cascade filter. The user can move the cascade sections in the order desired. After

each move of a cascade section, the roundoff power is found and presented to the user on the screen.

B.3.0.11 Help On How To Format Input Coefficient Files; Main Menu Option 11. This routine is built for first-time users of the system. This routine will allow the user to see the formats used in the direct form filter structures. This will show where the coefficients are located within the difference equation to avoid sign problems. This routine will also display to the user the formats used in the cascade form filters. Specifics are given so the user knows how to enter the coefficients and what to include with the cascade sections.

Also shown as screen outputs are examples of coefficient files for both the direct and the cascade sections. These examples will provide the user a means to immediately see the structure of the coefficient files. The examples will also give the user a means to avoid the pitfalls of wondering where the coefficients are located within the difference equation as well as the unique requirements on the cascade sections.

B.3.0.12 Running The Digital Filter Analyzer Software Tool; Main Menu Option "Return". This section is the main programming section where the computations are performed. The computations find the quantization effects on the filter in terms of magnitude, phase, and error. This part of the software is run by a user hitting the return key.

The code will take the information provided by the menu choices and perform the analysis accordingly. The user needs only to have a coefficient file in order to run this section. The result of running this section (referred to as running the program) is a set of five output data files. Two of these output files hold information about the filter done with single precision (unquantized) numbers. The other three output data files are pertinent to the quantized versions of the digital filter being analyzed.

Two files generated depict what would happen if single precision (unquantized) numbers are used for the filter simulation. Normally, 24-bits is adequate to maintain sufficient accuracy in the results. The two files are the magnitude and the phase of the filter. These

outputs are found under the file names mag##.dat and phase##.dat. The unknown numbers in the output file name is the user's choice of a two digit number to associate with the unquantized output files.

The second set of output files are pertinent to the quantized versions of the filter. The user will select the number of bits for the coefficients. Based on this information, the software tool generates three files. The first is the magnitude file. This file is the transfer function of the system if only B-bits are available. The file will have a name like magq##.dat. The unknown number is the user's choice of a two digit number to associate with the quantized files. The second file generated is the phase file. This file will show the effects quantizing the coefficients to a B-bit length word. The file will have a name like phaseq##.dat. This becomes important when linear phase is an issue in the design. Quantization will have an effect on the linear phase characteristics; how much effect can be seen by this output file. The third file generated is the error file. The error file will have a name with the format of error##.dat. The unknown number is the user's choice of a two digit number to associate with the quantized output files. The error is a linear error or a difference between the two files, unquantized magnitude and the quantized magnitude files.

After a user selects to run the software analyzer tool, the main menu is written back to the screen and the user is again allowed to make changes to the options.

B.3.0.13 Variables used in the Controlling Section. The following is a list of the variables in the main section of the program. They are listed here for anyone desiring to modify the code. By reviewing these variables, the code can be followed in terms of the operations that need to be done. The algorithms that I wrote can be a bit more troublesome to figure out unless the operation being performed is well understood and written out in detail. The code is well modularized, therefore, one can modify without too much difficulty.

After the comment section in the major block, declaration of variables is done. For clarity I include the variables used and what they mean.

- ERROR(2048) -this array holds the amount of linear error from the quantized and unquantized versions.
- XAXISSTEP(2048) -radian frequency used for the computation. This number is used to write out to data files.
- UNQUANTMAG(2048) -array to hold the unquantized magnitude, used to find the linear error.
- QUANTMAG(2048) -array to hold the quantized magnitude, used to find the linear error.
- PI -this is pi, 3.14159.
- MAG_H(2048) -array to hold the unquantized magnitude in dB.
- MAGQ_H(2048) -array to hold the quantized magnitude in dB.
- PHA_H(2048) -array to hold the unquantized phase response.
- PHAQ_H(2048) -array to hold the quantized phase response.
- A(512) -array to hold the y coefficients.
- B(512) -array to hold the x coefficients.
- AQ(512) -array to hold the y quantized coefficients.
- BQ(512) -array to hold the x quantized coefficients.
- GPDLAY(2048) -group delay at the radian frequency.
- GPDLAYQ(2048) -group delay from the quantized coefficients at the radian frequency.
- LOWER -this number is the starting point for the radian frequency.
- STEP -amount of increase in radian frequency for computations.
- RANGE -amount of radian frequency space to cover in the simulation.

- W -computed present radian frequency point to use in the simulation, increased by step.
- NUMBITS -number of bits available to represent the coefficients used in the digital filter.
- NPOINTS -number of spectral points to calculate across the bandwidth of the signal.
- FILTERTYPE -signal to tell program what type of filter is to be used in the simulation.
- CONTROL -controls the major flow of the program, takes on values that send the program flow into the subroutines to complete the simulation.
- NUM -complex number to represent the amount of spectral magnitude in the numerator at a specific spectral point.
- DEN -complex number to represent the amount of spectral magnitude in the denominator at a specific spectral point.
- H -complex number to hold the value for the transfer function at a specific spectral point. The complex values are used to compute the magnitude, phase and error values that are real numbers.
- ANS -byte value used to input a [Y]es or [N].
- CFLAG -flag to prevent the program from performing a simulation without first having an input data file for the coefficients.
- DATA(2048) -array to buffer the unquantized phase values.
- FNAME -name of file that holds or will hold the coefficient values.
- FILENAME -concatenates the ASCII variables into a name for the file used with the output procedures.
- FILTERNAME -will take on one of two values, direct form or cascade form.

- C2 -takes on the value of UNQFNAME the unquantized file number.
- FNUMBER -holds a two ASCII value to concatenate with the quantized output files of magnitude and phase.
- UNQFNAME -holds a two ASCII value to concatenate with the unquantized output files of magnitude, phase, and error.
- C4 -concatenates C1 with C2 with C4 and forms the filename.
- C1 -holds the ASCII value of 'mag' used with forming output file names.
- C3 -holds the ASCII value of '.dat' used with forming output file names.
- C7 -holds the ASCII value of 'magq' used with forming output file names.
- C5 -holds the ASCII value of 'phase' used with forming output file names.
- C6 -holds the ASCII value of 'phaseq' used with forming output file names.

B.3.0.14 Summary. The user's manual is presented to provide the approach in the choices available when using the digital filter analyzer tool. The program is split up into three major block. The first block covers the Main Option Menu that will serve as the central means for the user to input commands. The Main Computation Section For Computations is block two and Subroutines Called By The Main Option Menu And Main Computation Section is block three. Chapter IV has discussions on these last two blocks.

The software tool is self-teaching and menu driven. There's a help menu to choose from and helpful hints are given throughout the simulation process. As designers, we don't think of the problems of limited accuracy for representation of numbers. Our designs are built on infinite precision numbers. This tool will show the designer the expected output of the implemented filter as dictated by the restrictions from the hardware implementation. The designer can find the least amount of hardware required to maintain acceptable results.

Appendix C. *Additional Subroutines for the Digital Filter Analysis Software Tool*

C.1 Inputs to the Digital Filter Analysis Software Tool

The following input routines will be used most often. They serve to specify the precision and maintain organization in the output files.

C.1.0.15 Subroutine Readcoeff; Reads in the Coefficients From the Keyboard or From the Disk File System. The digital software analyzer must have some coefficients to implement a filter. There exists two distinct mechanisms to input to the software tool the coefficients: the keyboard or reading a data file. Help screens are provided that explicitly show where the coefficients are located in the transfer function to build a data file. Refer to B.2.2 for details.

C.1.0.16 Subroutine Bitsavailable; Pick the Length of the Register Words to Represent Coefficients and Intermediate Results. This simple routine receives from the user the number of bits that will be available to the digital software analyzer tool to represent the numbers within the digital filter realization simulation. The number of bits does include the sign bit. Therefore, the actual number of bits B is found by $(B + 1)$ bits entered to the software tool. All two's complement representations do include the sign bit.

Most designs will not require more than 16-bits to represent the transfer function. To find the least number of bits and still maintain the performance goals is unknown. By actual simulation, the answer to this question be found.

C.1.0.17 Subroutine Filenumber; Two Digit Number Appended to All Quantized Output Files. The user of the digital software analyzer tool can easily run five or more test cases against one set of coefficients. For example, a designer wants to see output for 16-bits,

14-bits, 12-bits, 10-bits, and 8-bits to represent the register word lengths. Even ten or more trial runs might typify a design process.

In order to keep the output files organized, a two digit number is appended to the following files when written to the disk drive: magq##.dat, phaseq##.dat, and error##.dat. These files show the results of the transfer function, the phase response, and the linear error between the single-precision (unquantized) transfer function and the quantized transfer function.

The user can assign any two digit number to each run corresponding to a different number of bits available for the register length. The two digit number can actually be any two ASCII characters. This way a designer can organize all of the output into their own separate files. When a designer has even seven transfer functions to test, the number of output files grow. For this example, say ten runs are done for each filter for bit values of 16 down to seven bits. That will result in seventy files generated by the program. Seventy files can easily become unmanageable unless some scheme is used. Therefore, a user can simply number each run will a separate two digit number. Each new multiple of ten can correspond to the next transfer function. So now the designer can easily recognize any file and the meaning of each file.

C.1.0.18 Subroutine Unquantfile; Two Digit Number Appended to All Single Precision Output Files. One other set of output files generated by this software tool concerns the transfer function and the phase response of the filter. Since the use of single-precision is applied to all variables, the software has 24-bits of accuracy to represent all numerics. The software tool will find the results of the filter if all 24-bits are used in the computations. This corresponds to the unquantized versions or single-precision.

The user will want to label each set of output files with a two digit number. This way with each new filter, the user can change this number and retain unique files describing each transfer function. Using the example of seven different transfer functions, a user could append multiplies of 10 (10, 20, 30, 40, 50, 60, and 70) to each of the filter's transfer function's set of

output. That way, the user will be able to immediately differentiate between the unquantized data files.

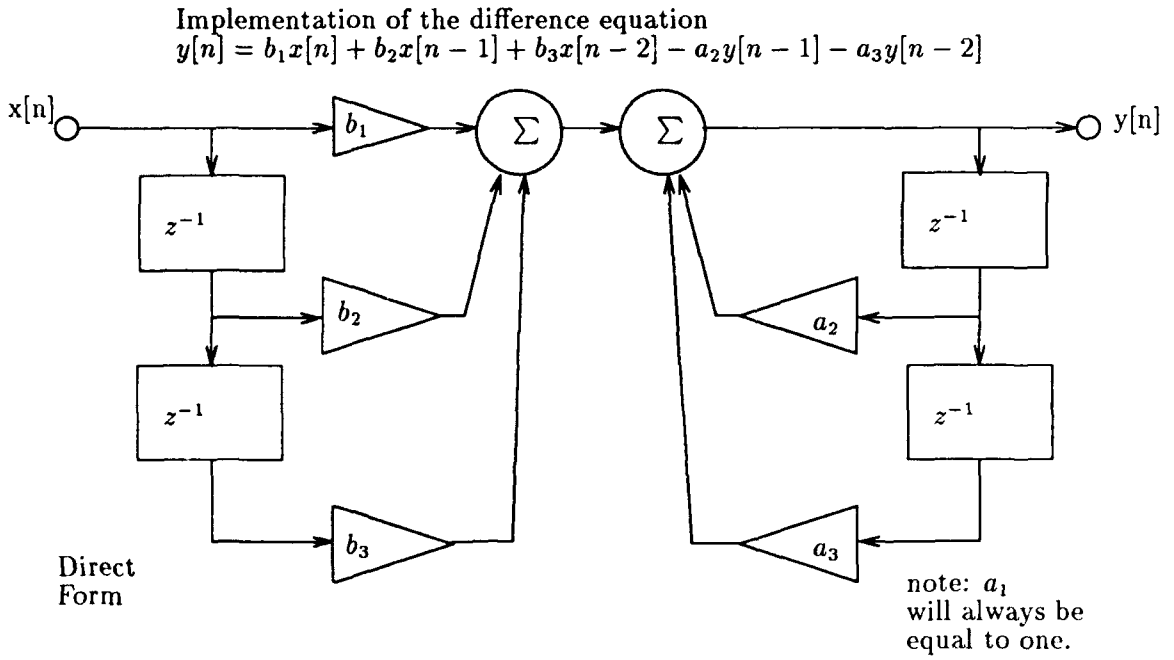
C.1.0.19 Subroutine Typeoffilter; Allows the User to Enter the Type of Filter, Either Cascade or Direct. This subroutine is called by the main menu. This subroutine also make subroutine calls to handle the conversion process.

There are two types of tasks performed by this subroutine. The first task prompts the user for the type of structure (direct and cascade) that will be analyzed by the digital software analyzer tool. The second task is not a straight forward process but rather a heuristic research tool that is designed for fine tuning the cascade structure.

The user of the software tool will have the choice between using the direct or the cascade structure as shown in Fig. C.1. When the user makes a selection between using one of the two structures, an example of the structure is sent to the screen along with other helpful information to keep the sign convention.

The second task is called conversion. Conversion involves taking the cascade structure of any order and converting (the process of conversion) the structure into a direct form structure. The result of the conversion process is the creation of a filter in a direct form. The conversion process does not change the position of the poles or zeros. It only changes the structure to implement the poles and zeros. The option to enter the conversion process is a separate subroutine. The conversion process re-builds a new filter structure and does allow the user to save the new set of coefficients for the direct design.

C.1.0.20 Subroutine Iterations; Enters the Number of Spectral Points to Evaluate. Resolution can be changed by entering the number of spectral points for the digital software analyzer tool to evaluate. The range in frequency is the normalized frequency band of $0.0 \rightarrow \pi$. This range in frequency is all the allowable frequencies without the effects of aliasing. Five hundred spectral points is the default value for the tool.



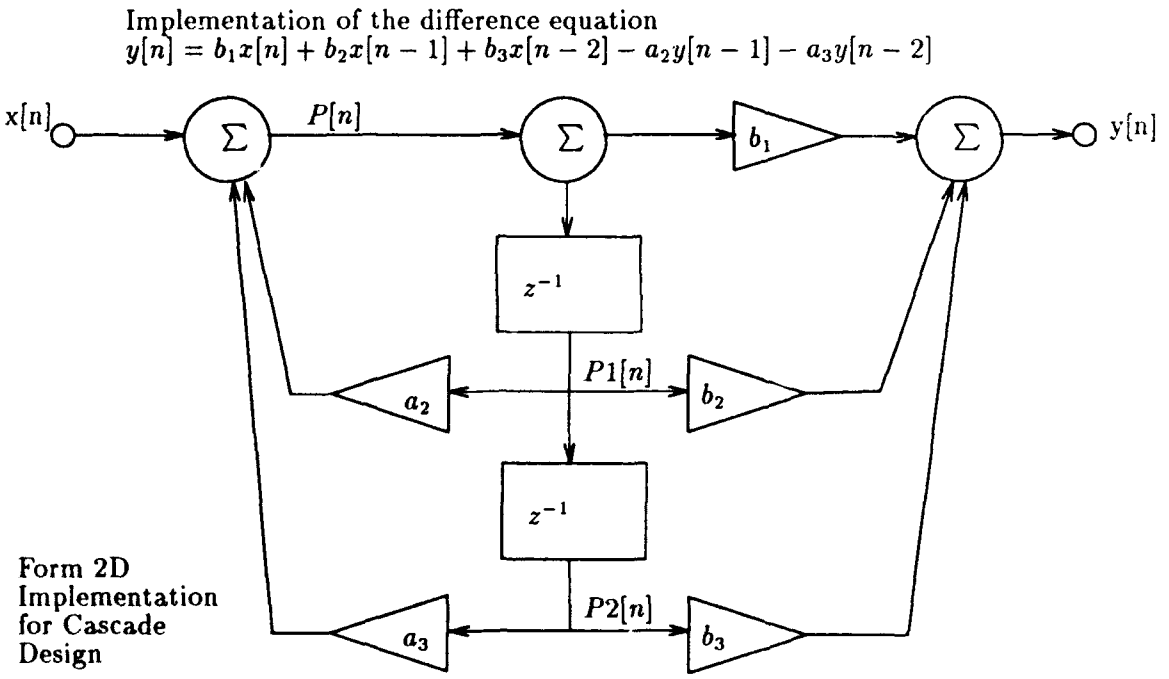


Figure C.1. Two types of structures analyzed by the digital software analyzer tool.

The output files are all based on the number of spectral points that are evaluated. A user can dramatically change the size of the needed disk space by changing the spectral points to 2500 points. Each file will take about 70,000 bytes for 2500 spectral points. The first run of the research tool will generate five files and each additional run will generate another three files. For example, if a user makes ten runs against one transfer function at 2500 spectral points, then the user must have 2,400,000 bytes available just to store the output generated. Typically, other files will be generated by the use of the data files for the generation of graphical data. Large amounts of disk space can be used by the implementation of this tool.

In order to cut down on the disk space, a user can change the number of spectral points. Even using the default value of 500 will more often than not provide adequate results and require only 15,000 bytes. In order to further reduce the needed space, a user can change the range of normalized frequencies to evaluate the transfer function as shown in Section B.3.0.7.

The user can experiment with these two adjustment values and use the number of spectral points in conjunction with the normalized bandwidth to view the pass band regions. This will often provide the needed detail in the critical areas of interest to the designer.

C.1.0.21 Subroutine Setrange; Enters the Normalized Bandwidth to Use for Output Files. The digital software analyzer tool uses a default range of $0.0 \rightarrow \pi$. A user can change this full available bandwidth by selection of main menu option seven. Nyquist limits the available frequency range to 1/2(sample rate). This limit is normalized to a range of values using π as the upper limit. The convention makes it convenient to take the transfer function's impulse response and translate it into any realization with its particular sample rate. Only then can the actual frequencies be known.

The user can input to the tool the lower and upper bounds for the normalized frequency bandwidth. The numbers entered are real values and in the range of $0.0 \rightarrow \pi$. The new range will be used by the tool in the evaluation of all the output files.

This option can provide detail for the user in the more critical regions. This will be in the areas where the passband is defined. Often, the user will find that by looking into the passband regions, the magnitude will not appear flat but contain oscillations. The data files generated will contain the spectral point where the item of interest was evaluated.

C.1.0.22 Subroutine Savecoef; Saves the Coefficients to a Disk File. The user will have opportunity to save coefficients to the hard disk file system to preclude from entering a set of coefficients time after time. The actual location of the output files will be where the user is running the application. The routine will initially use question marks to specify the file name. The length of the file name is limited to 15 ASCII characters.

All the routines that change the coefficients will use this subroutine. That way a user can re-write the coefficients to a file in the form that seems most suitable. When the structure changes as in the conversion process, then the user can have both sets of coefficients written to the disk, one in the cascade form and the other in the direct form. The form that this routine will save the coefficients in follows the convention that is required by the tool to read in from the disk system a coefficient file.

C.1.1 Subroutines for Structual Evaluation

C.1.1.1 Subroutine Conversion; Converts a Design Using the Cascade Structure Into a Design Using the Direct Structure. The conversion process is where the user can perform research on the implementation structure of a digital filter. This conversion utility will take the coefficients from the cascade structure and build a filter that uses the direct structure. The user can enter the conversion process by using main menu option 1. The conversion process will re-build a new set of coefficients. This re-building process is a destructive process. The new coefficients computed during the conversion will not be saved to disk. However, the user will be prompted to save the newly created coefficients to the disk system under a new data file name.

Table C.1. Forms of the difference equation in time, z-transform, and frequency.

Linear Constant Coefficient Difference Equation			
Signal Position	Coefficient	z-transform	Frequency Response
$y[n]$	1.0	$y(z)$	$y(e^{0j\omega})$
$y[n-1]$	a_2	$y(z^{-1})$	$y(e^{-1j\omega})$
$y[n-2]$	a_3	$y(z^{-2})$	$y(e^{-2j\omega})$
$y[n-3]$	a_4	$y(z^{-3})$	$y(e^{-3j\omega})$
$y[n-4]$	a_5	$y(z^{-4})$	$y(e^{-4j\omega})$
$x[n]$	b_1	$x(z)$	$x(e^{0j\omega})$
$x[n-1]$	b_2	$x(z^{-1})$	$x(e^{-1j\omega})$
$x[n-2]$	b_3	$x(z^{-2})$	$x(e^{-2j\omega})$
$x[n-3]$	b_4	$x(z^{-3})$	$x(e^{-3j\omega})$
$x[n-4]$	b_5	$x(z^{-4})$	$x(e^{-4j\omega})$
			where $\omega = 2\pi fT$

The algorithm is a two step process. The first step is the multiplication of two second-order polynomials. This step results in a fourth order polynomial. The second step is to take the second-order sections and continue to multiply them into the polynomial from the previous multiplication. This process keeps increasing the order of the polynomial by two with each additional multiplication process. The number of coefficients will increase by two as well for each additional multiplication step. The process to additionally multiply second-order sections is done by two inner do loops and one equation. The inner do loop runs through the second-order polynomial's coefficients. The outer do loop runs through the polynomial generated by the previous multiplication process.

C.1.1.2 Transfer Function Evaluation The method used to find the frequency response of the transfer function is to make the substitutions found in Table C.1. Euler's identity is used for the computation. The complex value is found at each spectral point according to the transfer function. The structure of the digital filter will change the method to compute the output of the transfer function. Hence, different structures of the same poles and zeros, produce different power spectral density functions.

C.1.1.3 Subroutine Directden and Subroutine Directnum; Evaluates the Transfer Function for a Direct Structure. The direct form structure's denominator and numerator is used to evaluate the complex value in frequency. This value is then used to find the magnitude and phase.

C.1.1.4 Subroutine Cascadenum; Evaluates the Numerator of the Transfer Function for a Cascade Structure. This subroutine takes the cascade form structure as a means to evaluate the transfer function. The coefficients for the numerator are passed into this subroutine. The complex value for the specific spectral point is found in the numerator. The routine will find the value of the transfer function for only one second-order numerator section. Then, depending on how many sections need to be evaluated, the next cascade second-order section's complex value is found. This value is then multiplied to the previous complex value found from the transfer function. This process continues until all second-order sections are computed.

This idea uses the principle that transfer functions in serial are simply equivalent to the transfer functions multiplied together. That means the system can be represented by transfer functions in series that hold only one second-order cascade section. These second-order cascade sections are multiplied together in the frequency domain to give a new transfer function. The value found from this routine is a complex value. This resultant value is used to find the magnitude and phase.

C.1.1.5 Subroutine Cascadeden; Evaluates the Denominator of the Transfer Function for a Cascade Structure. This subroutine uses the coefficients in the denominator of the cascade structure to evaluate the complex value for the specific spectral point. This routine is the same as in Section C.1.1.4.

C.1.1.6 Normalization; Normalizes the Coefficient File to a Magnitude of One. The user of the digital filter analyzer tool is encouraged to type in the coefficients as developed. These coefficients will often be of magnitudes greater than one. Whenever this

out of bound condition occurs during the running of the research tool, a flag is set and the user is presented with warning messages. The user will then have the option to run the normalization routine.

The normalization routine can be run on a set of coefficients that do not have a magnitude greater than one. When this is done, the user will be presented with the option to normalize the coefficients. Normalizing the coefficients in this case, expands the dynamic range of the digital filter. In terms of signal to noise ratio, the user effectively enhances this term. Analysis assumes the use of all bandwidth, but when only half of the range is in reality used within a digital filter then only half the signal to noise ratio really exists.

C.1.2 Sign Convention for Specification of Coefficients. Different books present the difference equations in various ways. Some may have the coefficients for the output terms on the left side as positive values while others may have the coefficients for the output terms on the left side as negative values. What convention is followed? The answers to sign convention can be easily found by reviewing the help routine found in the main menu. The conventions are repeated in two other areas of the program.

The method of sign convention is important since a sign error will certainly have dramatic effects on the shape of the magnitude plot. The user will find this convention stems from the difference equation that has the form of

$$\begin{aligned} y[n] &= b_1 x[n] + b_2 x[n-1] + b_3 x[n-2] + b_4 x[n-3] \cdots \\ &\quad -a_2 y[n-1] - a_3 y[n-2] - a_4 y[n-3] - \cdots \end{aligned} \tag{C.1}$$

where the coefficient $a_1 \equiv 1.0$.

When cascade second-order sections are part of the design criteria, a different form applies in the difference equation as shown in Eq. C.2. Since the use of second-order sections implies that the polynomials are limited to second-order, the difference equation takes on

the form

$$y[n] = b_1 x[n] + b_2 x[n-1] + b_3 x[n-2] + a_2 y[n-1] + a_3 y[n-2] \tag{C.2}$$

where each cascade section is built from the difference equation.

Higher order cascade filters are designed by putting together more of the second-order sections. The filter is simply a process of cascading second-order sections. The transfer function for the system would have the following form

$$H(z) = H_1(z) * H_2(z) * H_3(z) * \cdots \tag{C.3}$$

where each $H_k(z)$ takes on the form found in Eq. C.2. The order of the filter will depend on how many of the second-order sections are cascaded.

This appendix includes the subroutines that are needed to for the digital filter simulator tool. These routines are considered easier to implement. The program code can be better understood by reading the explainations given here.

Bibliography

1. Cooley, J. W. and J. W. Tukey. "'An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computing, 297-301*(1) (1965).

2. DeFatta, David J. and others. *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

3. Gabel, Robert A. *Signals and Linear Systems* (Second Edition). New York: John Wiley & Sons Inc., 1980.

4. Gold, B. and C. M. Rader. *Digital Processing of Signals.* New York: McGraw-Hill Book Company, 1969.

5. Jackson, L. B. "On the Interaction of Roundoff Noise and Dynamic Range in Digital Filters," *Bell Systems Technical Journal, 49*:159–184 (February 1970).

6. Jackson, L. B. "Roundoff Noise Analysis for Fixed-Point Digital filters Realized in Cascade or Parallel Form." *I.E.E.E. Trans. Audio Electroacoust.AU-18.* 107–122. New York: IEEE Press, June 1970.

7. Jackson, Leland B. *An Analysis of Roundoff Noise in Digital Filters.* PhD dissertation, Stevens Institute of Technology, Hoboken, N.J., 1969.

8. Jackson, Leland B. *Digital Filters and Signal Processing* (Second Edition). Boston, MA.: Kluwer Academic Publishers, 1986.

9. Kaiser, J. F. "Design Methods for Digital Filters." *Proceedings of the First Allerton Conference on Circuit and System Theory.* 221–236. 1963.

10. Kaiser, J. F. *Digital Filters Systems Analysis by Digital Computer.* New York: Wiley, 1966.

11. Kuc, Roman. *Introduction To Digital Signal Processing.* McGraw-Hill Book Company, New York: McGraw-Hill Book Company, 1988.

12. Lee, William N. *Minimization of Roundoff Noise in Digital Filters.* PhD dissertation, Standford University, Stanford, California, August 1971 (AD-730505).

13. Lee, William S. "Optimization of Digital Filters for Low Roundoff Noise." *Transactions on Circuits and Systems3: CAS-21.* New York: IEEE Press, May 1974.

14. Mendenhall and others. *Mathematical Statistics with Applications* (Second Edition). Boston: PWS Publishers, 1981.

15. Oppenheim, Alan V. and Ronald W. Schafer. *Discrete-time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

16. S., Chan David and Lawrence R. Rabiner. "Analysis of Quantization Errors in the Direct Form for Finite Impulse Response Digital Filters." *Transactions on Audio and Electroacoustics4: AU-21*. 255–265. New York: IEEE Press, August 1973.

17. Shanmugan, K. Sam and Arthur M. Breipohl. *Random Signals: Detection, Estimation, and Data Analysis*. New York: John Wiley and Sons, 1988.

18. Widrow, B. "Statistical Analysis of Amplitude-Quantized Sampled-Data Systems." *AIEE Transactions Applications and Industry81*. 555–568. New York: IEEE Press, January 1961.

19. Widrow, B and Samuel Stearns. *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1985.

20. Williams, Rob. Class Lecture. EENG791, Adaptive Signal Processing, AFIT, Wright Patterson AFB, OH, 15May 1991.